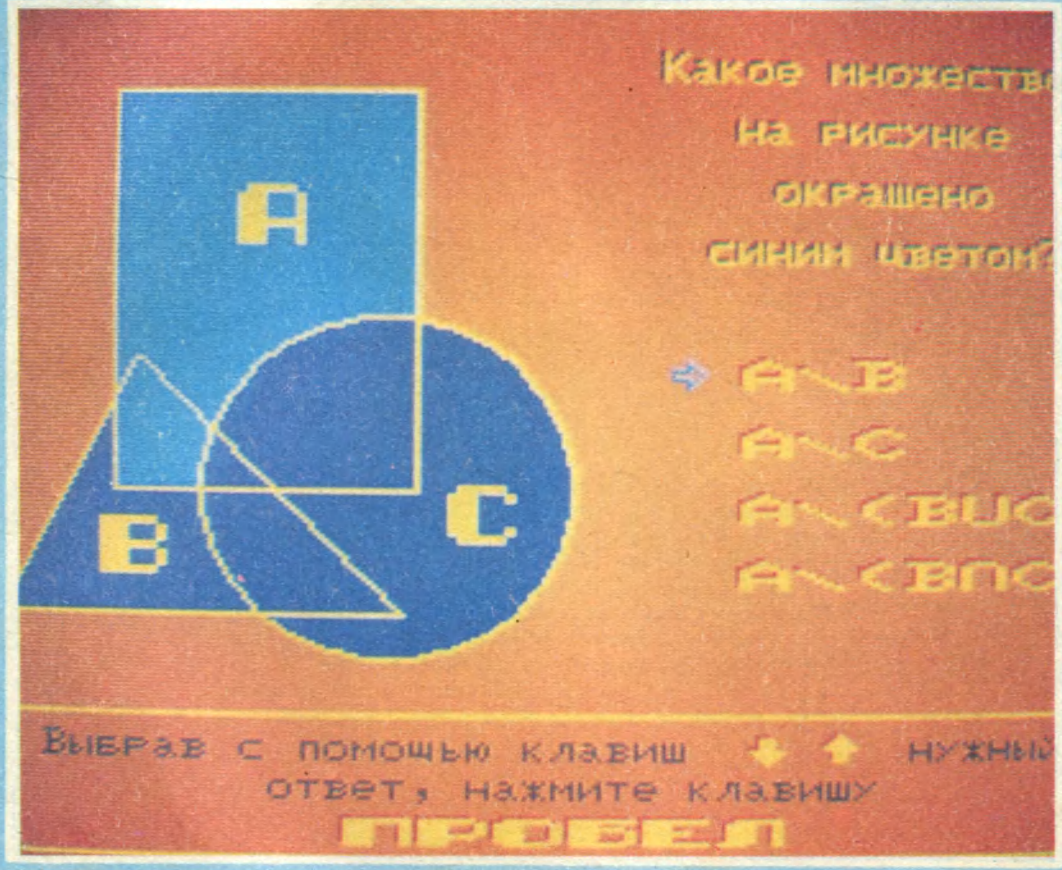
ISSN 0234—0453

ИНФО

ИНФОРМАТИКА И ОБРАЗОВАНИЕ

5 1990



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

5 '90
сентябрь — октябрь

Содержание

Педагогический опыт

Внеклассная работа

Зарубежный опыт

Нам пишут

Информация

Обложка Э. Бажилина

Редактор отдела *А. Кравцова*
Научный редактор *Т. Драгныш*
Зав. редакцией *Н. Игнатова*

Художественный редактор *Л. Розанова*
Корректор *Л. Яковлева*

Сдано в набор 25.07.90. Подписано в печать 30.08.90. Формат 70×100 1/16.
Бумага офсетная № 1. Печать офсетная. Усл. печ. л. 10,40.
Усл. кр.-отт. 42,88. Уч.-изд. л. 12,97.
Тираж 72 520 экз. Заказ 1312. Цена 60 коп.

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

Адрес для переписки: 107005, Москва, Лефортовский пер., 8.
Телефон редакции: 249-97-77 (или 261-11-29).

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

Информатику необходимо сохранить

В Гособразование СССР приходят многочисленные запросы от администрации и учителей школ по поводу существования курса информатики как самостоятельного школьного предмета.

Учитывая, что базисный учебный план (приказ Гособразования СССР от 22.09.89 г. № 751) регулирует лишь базовый компонент общего среднего образования без дифференциации на отдельно взятые предметы, предполагается, что эта дифференциация обязательно проводится на республиканском уровне.

Иными словами, распределение учебных часов между математикой и информатикой, объединенных в базисном учебном плане в один цикл учебных предметов, производится союзной республикой в зависимости от состояния материальной базы, подготовленности учителей и других особенностей региона.

В соответствии с Концепцией информатизации образования, утвержденной Коллегией Гособразования СССР от 01.12.89 г., курс «Информатика» является важным компонентом содержания общего среднего образования, в связи с чем **нельзя считать допустимым наметившуюся тенденцию к уменьшению отводимых на курс учебных часов или к исключению его из учебного плана школ.** В настоящее время необходимо сохранить курс в старших классах школы с целью получения первоначальной компьютерной грамотности в период школьного обучения. Содержание и уровень освоения курса могут быть различными в зависимости от выбранного школой профиля.

Введение углубленного изучения информатики находится в компетенции школы и осуществляется за счет часов учебного плана школьного компонента.

Н. НЕЧАЕВ,
начальник Главного учебно-методического управления
общего среднего образования, член коллегии

Районам и школам в настоящее время предоставлена значительная свобода в создании собственных вариантов учебного плана школы. В этой связи можно высказать в адрес руководителей народного образования несколько аргументов в пользу существования информатики как самостоятельного предмета.

1. Информатика — необходимый компонент среднего образования

При традиционных методах обучения основной целью являлось научение школьников определенным заниям, умениям и навыкам, теперь появляется возможность обучать технологиям. Обучение понятиям (know what) сменяется обучением технологиям (know how). Разумеется, обучение понятиям входит и в обновленный процесс обучения, но не как самостоятельная цель.

Каждый школьный предмет отличается своим, только ему присущим набором базовых понятий и технологий. Именно наличие таких базовых понятий и технологий делает обоснованным включение этого предмета в школьный курс. В связи с информатизацией образования можно выделить три базовые технологии — создание алгоритмов, создание информационных моделей, работа с базовым программным обеспечением ЭВМ — основным инструментом новых информационных технологий. Базисность третьей технологии в особых

объяснениях не нуждается. Базисность остальных двух объясняется следующим образом. Каждый школьный предмет включает изучение технологий составления алгоритмов решения соответствующих задач (математика — математических, физика — физических и т. д.). Существует много общего в составлении этих алгоритмов: например, метод нисходящего проектирования. Иначе говоря, существует общая технология создания алгоритмов, которая и конкретизируется в каждом из школьных предметов. Естественно, эта технология должна быть изучена школьниками в «чистом» виде. Точно так же каждая наука, а значит и каждый школьный предмет, занимается составлением информационных моделей жизненных задач. Отличие между науками состоит в различии методов и средств построения моделей. В то же время построение моделей во всех науках имеет много общего; существует единая базовая технология составления информационных моделей, которая должна быть изучена отдельно.

Сказанное выше означает, что должен существовать отдельный школьный предмет, содержанием которого было бы изучение указанных трех технологий. Такой предмет есть — «Информатика». Этот предмет не может быть растворен в других, поскольку остальные предметы изучают лишь конкретизации данных технологий. Он должен изучаться в базовой школе, т. е. до или во время того, как конкретизации трех упомянутых технологий будут изучаться в других предметах. Особая роль предмета «Информатика» состоит еще и в том, что он сравнительно молод и создание соответствующего компьютерного курса не требует болезненной ломки сложившихся методик (как это неминуемо случится с другими предметами).

2. Созданы условия преподавания информатики

В течение 1985—1990 гг. подготовлены десятки тысяч квалифицированных учителей информатики, имеющих уникальный опыт преподавания курса по машинному варианту. Во многих школах эти учителя взяли на себя функции организаторов компьютеризации учебного процесса в целом. Отмена курса информатики повлечет за собой отток этих кадров из школ.

3. Создана методика преподавания информатики

Разработаны и внедряются вариативные методики преподавания курса информатики по машинному варианту, они обеспечены учебно-методическими комплексами, включающими в себя учебник, методическое пособие, задачник и программное обеспечение.

В содержании и методике преподавания информатики заложено решение задач по математике, физике, химии и другим предметам (в том числе и гуманитарного цикла), что обеспечивает постепенное внедрение методов информатики в другие общеобразовательные предметы.

Внедрение интегративного курса «Математика и информатика» требует коренного пересмотра содержания математического образования, разработки учебников и методик преподавания, массовой переподготовки учителей математики. Целесообразность такого курса должна быть доказана путем экспериментальной проверки.

Кроме того, нельзя сводить информатику к математике или даже к естественнонаучной проблематике, поскольку гуманитарный аспект информатики не менее важен.

4. Информатика — катализатор новых форм и методов обучения

Информатика меняет психологическую обстановку в школе. В процессе совместной деятельности по разработке учебного программного обеспечения реализуется педагогика сотрудничества учителя с учащимися. Существенно, что в такую деятельность постепенно вовлекаются и учителя предметники в качестве разработчиков методически обоснованных сценариев.

В информатике реализуются также новые подходы к контролю знаний и

умений учащихся. Вместо традиционных контрольных работ, нацеленных прежде всего на выявление степени механического запоминания учащимися учебного материала, проводятся тематические и итоговые зачетные работы, которые ориентируют учащихся на новые информационные технологии получения знаний и принятия решений. Учащийся может (и должен!) в процессе выполнения задания пользоваться различными источниками информации (учебники, справочники, а в перспективе — компьютерные справочно-информационные системы). Таким образом, учащийся готовится к функционированию в информационном обществе.

5. Учащиеся хотят изучать информатику

Информатика является одним из немногих предметов в школе, которые хотят изучать учащиеся. Этот интерес имеет как объективные (социально-экономические), так и субъективные (психологические) корни, и его нельзя игнорировать. Информатика является тем предметом (рычагом), который может позволить изменить отношение учащихся к учебе вообще.

Показателем такого интереса, в частности, являются пожелания учащихся многих школ включить информатику в перечень выпускных экзаменов. Целесообразно разрешить школам проводить выпускной экзамен по информатике.

Н. УГРИНОВИЧ,
зав. кабинетом информатики МГИУУ,
Л. САМОВОЛЬНОВА,
специалист I категории
Главного учебно-методического
управления общего
среднего образования
Гособразования СССР

Л. НЕВУЕВА, Т. СЕРГЕЕВА

О перспективных тенденциях разработки педагогических программных средств

В сегодняшней зарубежной и отечественной практике разработки компьютерных программ выделяется одно из наиболее перспективных направлений — проектирование инструментальных педагогических средств (ИПС). Это направление обязано своим возникновением новой полифункциональной учебной среде, органичной частью которой становятся новые информационные технологии. Оно ориентировано прежде всего на качественное изменение характера взаимодействия с компьютером двух основных групп пользователей: педагогов и учащихся.

В настоящей статье предпринимается попытка описания этой новой перспективы, дается рабочее определение содержания понятия «инструментальные педагогические средства», обозначается их место в ряду других педагогических программных средств (ППС), анализируются условия возникновения устойчивой тенденции к разработке подобных программ, их отличительные свойства и функции.

В соответствии с целевым назначением инструментальные педагогические средства включают в себя широкий спектр разнообразного методического оснащения учителя, а также программы, рассчитанные на самостоятельную индивидуальную или коллективную работу школьников.

Определение ИПС с необходимостью требует обращения к рассмотрению предшествующего периода разработки ППС. Проведенный анализ показывает, что существующий банк программ может быть разделен на две условные группы,

которым соответствуют фактически противоположные направления использования ЭВМ в обучении: программированное обучение и обучающие технологии компьютерных предметно-ориентированных сред.

Основные теоретические ограничения программированного обучения, заключающиеся в ориентации на рецептурно-процедурную и, главным образом, репродуктивную активность обучаемого, составили концептуальную базу громадного большинства обучающих программ, обеспечивающих тренинг типовых умений и ознакомительную форму представления знаний. Статичный характер подобных учебных средств, практически полностью повторяющих идеологию и технологию пособий по программированному обучению, исключает возможность продуктивного освоения учащимися конструктивных способов деятельности при формировании умений и навыков, развитии способностей, овладении понятийным предметным содержанием.

Надо сказать, что, с точки зрения истории развития информатизации, осмысление первого этапа применения вычислительной техники и технологии в сфере образования позволяет понять очевидный бихевиористский акцент первых (как, впрочем, и многих последующих) попыток проектирования обучающих программ, поскольку начало внедрения ЭВМ в учебный процесс связано с лимитированными возможностями машин и естественным техноцентризмом и даже «технороматизмом» профессиональных программистов, по необходимости ставших авторами и разработчиками нового педагогического инструментария и не имевших при этом ни подходящих аналогов и прототипов, ни специальных знаний.

Идеология классического программированного обучения, отразившаяся, как в зеркале, в соответствующих ориентациях первых разработчиков обучающих программ, не смогла обеспечить теоретически обоснованную психолого-педагогическую перспективу развития той новой реальности информационных обществ, которая выразилась в появлении «новых информационных технологий обучения». Действительность потребовала сначала смены самой тактики, т. е. широкого привлечения разнообразных специалистов по обучению, а затем и стратегии разработки ППС.

По ряду объективных обстоятельств (недостаточный парк машин, обеспечивающий 1 % компьютеризации процессов производства, связи, информационного обмена и обучения по сравнению с развитыми странами, отсутствие квалифицированных специалистов по проектированию новых учебных технологий, широко финансируемых национальных и целевых региональных программ информатизации общества и сферы образования и т. д.) в нашей стране очерченный выше период не сопоставим с зарубежным, исключая длительный и недостаточно эффективный отечественный опыт разработки АОС. Этот период фактически включает в себя последние 5—7 лет, которые завершаются уже формулируемой в настоящий момент потребностью в конструктивных теоретических позициях, предваряющих разработки компьютерных обучающих программ.

Аналогичная тенденция прослеживается и за рубежом, где за последнее десятилетие сформировалось новое направление в теории и практике использования ЭВМ в целях обучения, которое связано с идеями «конструктивизма» (С. Пейперт). Основная посылка конструктивистской модели С. Пейперта, включающей в себя кроме когнитивного подхода к проблемам обучения в духе Пиаже также и мотивационную сторону процесса, заключается в необходимости создания специальной обучающей среды («среды ЛОГО»), естественно ориентирующей детей на осмысленное и самодеятельное применение компьютера. Сутью развиваемой Пейпертом и его последователями перспективы является целенаправленное продуцирование условий, благоприятных для спонтанного возникновения интеллектуальных моделей в мышлении ребенка, когда по мере развертывания процесса «прорастания» одних из других последние приобретают логическую и эмоциональную форму. Анализируя опыт построения «Черепашьей геометрии» — компьютерного варианта введения детей в математические понятия, Пейперт формулирует

свое представление о новых функциях образования в информационном обществе, которые заключаются в том, что организация процесса обучения, и прежде всего процесса компьютерного обучения, требует создания специальной обучающей среды в виде особых предметных «микромиров»: «страны математики», «страны физики», «мира письменного языка» [6], т. е. таких же естественных контекстов для изучающих школьные дисциплины, каким в идеале является для каждого ребенка родноязычный языковый социум. Главное назначение предметных сред — побуждать ребенка активно участвовать в процессе конструирования «объектов-с-помощью-которых-можно-думать» («objects to think with» — там же. С. 128). Это совершенно особые объекты, в которые сведены в единое целое признаки культурной среды, «встроенное» значение и возможность персональной идентификации в процессе обучения. Последнее означает, что, задавая поведение объекта и размышляя над этим поведением, ребенок осознает свои собственные шаги и намерения и это осознание своего поведения выступает для него как источник содержательных идей для общения с Черепахой и как основание для выбора адекватных управляющих воздействий, что невозможно без сознательного планирования собственных действий.

Согласно Пейперту общение с компьютером в интерактивном режиме, протекающее в форме увлекательной игры, которая требует пространственно-временных средств коммуникации и способствует их развитию, не может быть сведено к простому удовольствию. Это «умственное упражнение» позволяет детям, работающим с электронным графопостроителем, «редактором цвета», «лингвистическим редактором», осваивать особый метаязык, на котором можно рассуждать о формах, их постоянном изменении, о скоростях, процедурах и т. д. Иными словами, дети осваивают формальный язык и одновременно с этим конструируют новый образ понимания самих себя, в котором естественно слиты рациональные, интуитивные, чувственные и эмоциональные компоненты.

Анализ теоретических оснований обучающей системы ЛОГО, подтверждающей роль действий самого ребенка в процессе обучения, показывает, что наиболее передовые подходы в зарубежной психологии к проблеме компьютерного обучения созвучны деятельностным концепциям психического развития, разрабатываемым в научных школах Л. С. Выготского, А. Н. Леонтьева, В. В. Давыдова. С другой стороны, в последнее время появилось немало работ [4, 7], в которых центральная идея С. Пейперта о спонтанности действий учащихся подвергается критике с точки зрения требований к организации среды обучения, соотношения целей учителя и учащегося, способов координации обучаемого и того, кто является носителем социокультурных норм.

Наряду с этим дебатируется вопрос о критерии «свободы» — степени необходимости или желательности спонтанности. Согласно Ж. Пиаже истинным можно назвать познание, основанное на самостоятельном открытии, однако рассчитывать на то, что учащийся способен случайно и непроизвольно воспроизвести достаточный для своего будущего объем исторически накопленного опыта, не приходится, тем более, что современная культура почти полностью лишает детей непосредственного материала в виде промежуточных объектов, из которых и на основе которых можно строить модели рассуждения, математические, физические и другие понятия и идеи. Отсюда следует, что адекватно построенное обучение, в котором спроектирована логика построения объекта усвоения, его «жизнь» и становление, а это становление, в свою очередь, дано через действия и взаимодействия обучающих и обучаемых, не только не ограничивает, но, напротив, расширяет свободу ребенка и создает условия для его развития. Исследователи, рассматривавшие вопрос о целесообразности структурирования и руководства деятельностью обучаемых по использованию ЛОГО [5], пришли к выводу, что хорошее методическое пособие не подавляет, а скорее стимулирует самостоятельность учащегося. При правильном применении оно становится частью среды, с которой взаимодействуют дети, обеспечивая им большую воз-

можность достижения целей без помощи преподавателя. В этом смысле выдвинутое Пиаже понятие обучения через открытие к настоящему моменту эволюционизировало в сторону «направляемого обучения через открытие».

Такая тенденция в развитии отечественной практики применения компьютера для целей обучения отразилась в возникновении качественно новой ориентации — разработке инструментальных педагогических средств. Она связана, с одной стороны, с попытками аккумулировать предшествующие эмпирические результаты, а с другой — максимально эффективно и целенаправленно использовать достижения психолого-педагогических наук при проектировании ППС. ИПС по своему назначению входят в более широкую группу средств, названных ранее педагогическими программными средствами (определение ППС можно найти в кн.: Рекомендации по проектированию педагогических программных средств — психолого-педагогический аспект: Материалы III заседания Рабочей группы по методологии разработки ППС. София, 1988), и не имеют отношения к специальным профессиональным «инструментальным средствам», используемым программистами.

Исходно в группу ППС входят и собственно обучающие программы в традиционном смысле слова. Они характеризуются, как правило, жестко заданным алгоритмом исполнения программы, в структуру которой заложена одна неварьируемая методика освоения содержания. Попытки трансформации или адаптирования подобных программ приводят к их практическому разрушению. Сказанное не означает, что от программ такого типа следует отказываться, напротив, их необходимо совершенствовать с точки зрения более психологически выверенного построения предметного содержания и предлагаемых к освоению действий, а также целесообразности использования в конкретных учебных ситуациях. Т. е., правильнее было бы говорить не о противопоставлении ИПС обучающим программам, а об их развитии в направлении большей адаптивности, интерактивности и дидактической полифункциональности. Обучающие программы в пакете ППС призваны нести информационную и методическую нагрузку в условиях, когда объективно испытывается дефицит в профессионально подготовленных кадрах, методических пособиях, обеспечении грамотной индивидуализации и дифференциации обучения (допустим, в малокомплектных сельских школах). Значимость таких «стационарных» программ, способных обеспечить воспроизводимость определенных педагогических эффектов как для учащихся, так и для учителей, трудно переоценить (2, 3).

Вместе с тем принципиальное отличие ИПС от обучающих программ заключается, по нашему мнению, в следующем: они позволяют учителю осуществлять обучение на качественно более высоком уровне использования конструктивно-комбинаторных возможностей используемых средств;

ИПС не обязательно требуют создания принципиально новой (чисто «компьютерной») методики, а предполагают органичное сочетание привычных форм и приемов работы с новационными подходами и способами, создавая среду для расширения методического инструментария учителя, а не его ломки;

ИПС по определению имеют широкий спектр применения, отличаются инвариантностью и модульностью, что обеспечивает возможность практически каждому педагогу независимо от опыта и методической оснащенности творчески реализовать свои индивидуальные возможности в построении стратегии и тактики обучения.

Обратимся к описанию функций ИПС. Отличительной особенностью ИПС выступает их полифункциональность. Фактически следует говорить о нескольких группах функций: справочно-информационных и вычислительных; коммуникативных (обеспечение интерактивности, диалоговости, форм и способов связи, адаптивности) и конструктивно-комбинаторных.

Носителями первой группы функций — справочно-информационных и вычислительных — являются ИПС обеспечивающие хранение больших объемов информации в структурированном виде и оперативный пользовательский

доступ к ней; сюда же относятся банки данных, базы знаний, телетекст, экспертные системы и т. п. Справочно-информационная функция реализуется в избирательном, интерактивном режиме и информация предоставляется по сервисному принципу, т. е. пользователю не разрешается введение новой информации [1]. Конкретным примером реализации обсуждаемой функции могут служить экспертные системы педагогической и психологической диагностики. В экспертных системах подобного рода входной информацией могут быть характеристики, описывающие поведение ребенка в конфликтной ситуации и запрос о тактике действий педагога в этом конфликте. Выходными сведениями в этом случае становится информация для принятия решения преподавателем. Другим примером может служить база данных, описывающая передовой педагогический опыт, хранящая реестр новых методических разработок, пособий, рекомендаций и т. д. Для иллюстрации вычислительных функций можно указать не существующие и проектируемые ИПС по автоматизированной, статистической обработке данных, например физического или химического эксперимента. Реализация функций данной группы осуществляется за счет профессионально-ориентированного языка, который позволяет пользователю без труда входить в любую базу данных, экспертную систему, не обращаясь при этом к специальному языку программирования.

Вторая группа функций включает все формы коммуникаций с помощью ЭВМ и направлена на обеспечение организационных форм обучения, выбор режимов общения и взаимодействия (с машиной, учителем, сверстником, внутригруппового, межгруппового и т. д.), трансляции предметного содержания и связи между всеми участниками учебного процесса. Благодаря коммуникативной функции ИПС пользователь получает возможность видоизменять как свою деятельность, так и деятельность того, с кем он взаимодействует и наоборот. Примером реализации данной коммуникативной функции могут служить учебно-игровые средства организации конкретных форм учебных ситуаций, которые построены по принципу того или иного типа взаимодействия: парное взаимодействие, «восходящие» и «нисходящие» программы или модули программ, обеспечивающие связь учителя с учащимися и наоборот, внутри- и межгрупповая коммуникация, электронная почта, телеконференции и др. Качественный уровень коммуникативности ИПС порождает соответствующие условия возникновения внутренне мотивированной и целесообразной деятельности для всех участников учебного процесса, их естественной вовлеченности в освоение предметного содержания. Обеспечение такой функции в ИПС снимает проблему внешней занимательности, искусственно и формально поддерживающей интерес и активность обучающихся.

Последняя группа функций связана с реализацией конструктивно-комбинаторных возможностей ИПС как средств представления предметного содержания. Собственно благодаря этой группе функций и обеспечивается «жизнь» предметной среды как мира объектов, с которыми работает учащийся. Мера адаптивности ИПС находится в прямой связи с их конструктивно-комбинаторными возможностями, которые, в свою очередь, обеспечиваются полнотой всего комплекса элементов среды, системностью их связей, широкой вариативностью материала, потенциальных действий и операций, средствами преобразования одних в другие, наличием спектра моделей и схем вместе с приемами их трансформации, свертывания и развертывания и т. д.

Проектирование ИПС с означенными группами функций представляет собой многоплановую задачу, которая не может быть решена без интегрированных усилий специалистов высокого уровня и различной специализации. Однако объединение сил педагогов, методистов, психологов, программистов требует предварительной концептуальной согласованности относительно принципов построения той деятельности, частью которой может стать работа учащихся и учителей с ИПС, и, исходя из этих принципов, определение самого понятия инструмен-

тальных программных средств. В данной статье мы попытались дать рабочее и по необходимости предварительное описание исходных оснований и функций ИПС в учебном процессе. Практическое проектирование и апробация программ типа ИПС помогут раздвинуть и уточнить границы этого нового понятия.

Литература

1. *Франс А.* Заочное обучение и коммуникация с помощью ЭВМ // Перспективы. 1989. №. 1. С. 82—89.

2. *Вильямс Р., Маклин К.* Компьютеры в школе. М.: Прогресс, 1988.

3. *Сьюэлл Дэйв Ф., Ротерей Дейвид Р.* Основные направления применения ЭВМ // Перспективы. 1988. № 3. С. 60—69.

4. *Corte E.* De Verschaffel L. Logo: A Vehicle for Thinking: Report // Center for Instructional Psychology. Belgium, 1987. 27 p.

5. *Leron U.* LOGO Today: Vision and Reality // The Computing Teacher. 1985. February. P. 26—32.

6. *Papert S.* Mindstorms: Children, Computers and Powerful Ideas. Brighton, 1980.

7. *Dea R. De, Kurland M., Hawkins J.* LOGO and the Development of Thinking Skills. // *CHEN M. and Paisley W.* (eds.). Children and Microcomputers: Research on the Newest Medium. Beverly Hills, 1985. P. 193—212.

В. БОЛТЯНСКИЙ, В. РУБЦОВ

Игровые компьютерные среды учебного назначения

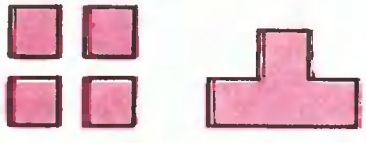
Сейчас многими фирмами (особенно западными) в огромных количествах тиражируются компьютерные развлекательные игры. Ребенок (а подчас этим увлекаются и взрослые) азартно гоняет по экрану корабли, ракеты, фигурки полицейских, стремясь что-либо взорвать, догнать, от кого-либо убежать и т. п. Некоторая польза от таких игр есть. Например, ребенок привыкает к клавиатуре компьютера, развивает быстроту реакции. Некоторые «деловые» игры учат принимать решения, ориентироваться в разных ситуациях. И все же в целом азартные развлекательные игры вредны, поскольку приводят к бездумно-расточительной трате времени.

Вместе с тем существуют компьютерные игры другого плана — развивающие, интеллектуальные. Их роль очень велика [1, 2]. Они направлены на развитие логического мышления, навыков исследовательской работы, творческого потенциала.

В этой статье мы изложим свои взгляды по этой проблеме и расскажем о некоторых компьютерных развивающих играх.

Опишем первые четыре игры разработанного авторами пакета компьютерных развивающих игр «Геометрическая комбинаторика». Всего он содержит 6 игр, предназначенных для учащихся младших классов и имеющих своей целью развитие геометрических представлений, а также первоначальное знакомство с понятием площади.

В первой игре пакета «Геометрическая комбинаторика» рассматриваются задачи, связанные с заполнением различных фигур квадратиками. Вначале предлагаются фигурка, имеющая форму перевернутой буквы «Т», и четыре отдельных квадратика (рис. 1). Довольно нетрудно, мысленно разделив эту фигурку на квадратики, понять, что этими четырьмя квадратиками можно предложенную фигурку покрыть. Программа составлена так, что первый раз действие производится автоматически, причем текст в нижней части экрана комментирует происходящее. Ребенок знакомится с термином «курсор» (здесь он имеет вид стрелочки) и с его перемещением по экрану при помощи клавиш, на которых обозначены стрелки. Он



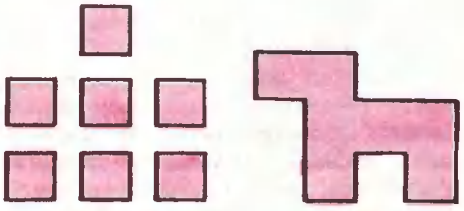
1

видит, что можно «захватить» квадратик, подведя к нему курсор и нажав клавишу «пробел», а затем можно переместить захваченный квадратик и (с помощью той же клавиши «пробел») оставить его в нужном месте. Теперь ребенок знает, какими клавишами он может оперировать.

Это упражнение несложно, но оно интересно и, главное, позволяет ребенку действовать самостоятельно. Вслед за этим дается первоначальное представление о площади (для фигур, которые можно составить из единичных квадратиков).

Теперь ребенку одна за другой предлагаются несколько фигурок, которые надо заполнять квадратиками. Например, одна фигурка напоминает собачку, а в левой части экрана имеются семь квадратиков (рис. 2). Если мысленно расчленить фигурку на квадратики, то станет понятно, что она как раз составляется из семи единичных квадратиков, т. е. заполнение возможно. Следующие фигурки посложнее, причем после выполнения игры с фигуркой идет короткое обсуждение причин, вследствие которых возможно или невозможно составить эту фигурку из заданных квадратиков: надо, чтобы число данных единичных квадратиков было равно площади фигурки (т. е. числу квадратиков, на которые можно мысленно расчленить эту фигурку).

2



В конце игры ребенку предлагается самому построить какие-либо фигурки из квадратиков. Это способствует проявлению элементов творчества и конструирования, развивает геометрическую интуицию. Эта игра не очень сложная, но она учит элементам логического рассуждения и геометрическому воображению.

Вторая игра несколько сложнее. Здесь ребенку предлагается составлять фигурки из «кирпичиков» двух видов: квадратиков и их «половинок» — треугольников. При этом ребенок должен сам выбрать, какое число квадратиков и какое число треугольников он хочет взять. Разумеется, треугольников надо взять четное число (0, 2, 4, 6 и т. д.), поскольку из двух треугольников можно составить квадратик. При этом для составления квадратика из двух треугольничков надо один из них повернуть (это осуществляется с помощью обусловленной клавиши). Кроме того, площадь треугольничка равна 1/2, а площадь квадратика — 1. Это позволяет подсчитать, сколько можно взять треугольничков и квадратиков (рис. 3). Если выбрано неправильное число треугольничков и квадратиков, постепенно выяснится, что ими заполнить фигуру не удастся (или же останутся лишние «кирпичики»). Все это помогает усвоению понятия площади, а также развитию геометрической интуиции: ребенок должен заранее представить себе, нужно ли треугольничек поворачивать, как прикладывать «кирпичики» друг к другу, сколько их взять и т. д.

Заметим, что эта игра допускает (как и другие игры этого пакета) не только индивидуальное, но и коллективное использование. Например, первый играющий указывает количество треугольников, а второй — недостающее число квадратиков. Первый передвигает (поворачивая, если нужно) лишь треугольники, а второй — квадратики. Возможна и конфликтная форма игры: поочередно размещаются треугольники и квадратики, а тот, кому уже не удается разместить свой элемент, считается проигравшим.

Третья игра, связанная с заполнением фигурок кирпичиками в форме «уголков», более сложна. Она геометрических знаний не требует, но геометрическая интуиция и пространственное представление постепенно развиваются,

3

причем «уголки» требуется поворачивать при прикладывании их друг к другу. Эта игра полезна не только для закрепления понятия площади, но также для развития логических навыков, творческого мышления, сообразительности и геометрической интуиции.

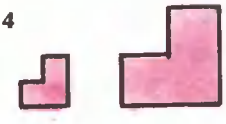
Очень интересно, в частности, заполнение «уголками» фигуры, которая составлена из трех больших квадратов (рис. 4). Так как площадь «уголка» равна 3, а площадь предложенной фигуры равна 12, то ясно, что «уголков» надо 4 штуки. Но чтобы решить, как уложить «уголки» для заполнения этой фигуры, требуется использование поворотов и геометрическое воображение.

Четвертая игра интересна тем, что здесь задаются все более сложные «кирпичики», но сложить из них надо прямоугольник (или квадрат). Ребенок сам должен решить, какой прямоугольник он сможет составить и сколько предложенных «кирпичиков» для этого нужно взять. Заметим, в частности, что из «уголков» (площади 3) можно составить прямоугольники 3×2, 6×2, 6×5, 6×6 (рис. 5). Общий рецепт таков: если каждая из сторон прямоугольника имеет длину не меньше 2, а произведение длин сторон делится на 6, то прямоугольник составить можно. Точно так же, если каждая из сторон имеет длину не меньше 4, а произведение длин сторон делится на 3, прямоугольник составить можно (рис. 6). Других случаев нет. (Здесь за единицу длины принимается сторона маленького квадратика, т. е. такого, что «уголок» составляется из трех таких квадратиков.)

Заметим, что более сложные «кирпичики» (как, например, на рис. 7) требуют для составления прямоугольников из них хорошо развитого геометрического воображения. Эта часть пакета развивающих игр может быть интересна и для старших школьников. Вообще игры «Геометрической комбинаторики» вызывают интерес у людей разного возраста, даже у взрослых.

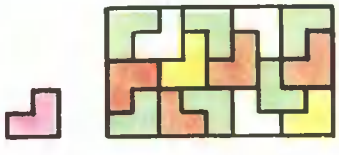
Таким образом, наша гипотеза состоит в том, что в основу проектирования компьютерных игровых сред учебного назначения должен быть положен принцип моделирования деятельности, в которой воссоздаются условия для поиска, отображения в моделях и анализа содержания объекта усвоения. Так, в нашем примере моделируется деятельность, позволяющая ученику осуществить анализ понятия площади. Компьютер выступает при этом в качестве средства моделирования и предметного содержания объектов усвоения, и обобщенных способов действия (определяющих решение задач некоторого класса), и организации совместной содержательной учебной деятельности (типа «обучаемый — группы учащихся», «ученик — ученик», «учитель — ученик»). Существенно при этом также, что компьютер используется в этих учебных ситуациях также как адекватное средство контроля и оценки действий учащихся, осуществляемых в процессе решения задач. И здесь важны не только накопление сведений о результатах, достигнутых учащимся, и статистическая обработка этих сведении с целью оценки знаний, полученных обучаемым, но также адаптивные функции компьютера, как, например, усложнение заданий при успешном решении нескольких задач, упрощение заданий при наличии ошибок (с возможным предоставлением повторительных упражнений по ранее не усвоенному материалу («тренажер») и даже имитация изучения психологии обучаемого.

4

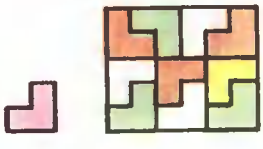


5

6



7



При проектировании учебных компьютерных сред требуется тщательно отбирать рекомендуемый материал [5]. Так, составлению пакета развивающих компьютерных игр предшествовал отбор материала, связанного с первоначальным понятием площади, и его анализ, в результате которого были отобраны рекомендуемые к усвоению учебные действия (составление фигур из квадратов и других элементов, подсчет числа составляющих частей, мысленное расчленение фигуры на части, комбинаторное мышление — перегруппировка частей, их расположение, развитие геометрического воображения).

Проектирование учебных сред включает в себя выявление и описание области знания, охватывающей конкретный объект изучения, а также определение его элементов и их отношений. Сюда же относится определение типов связей между элементами (возможно, в аксиоматическом описании). Эти связи характеризуют объект как некоторую систему.

Технология разработки компьютерных учебных сред включает в себя определение типа знаковых средств, обеспечивающих реализацию всех связей на дисплее, а также выбор динамического типа моделей, позволяющих изменять, трансформировать и конструировать объект изучения (предметные, графические, знаково-символьные и прочие модели). Сюда же относится определение и описание системы задач, обеспечивающих освоение объекта через его моделирование. Так, в нашем примере кратко описана и система задач (детально разработанных и включенных в пакет игровых программ, описанных в этом примере). При этом проблематика задач выбирается таким образом, чтобы, с одной стороны, исключались или существенно ограничивались эмпирические стратегии поиска («пробы и ошибки»), а с другой — обеспечивалась доступность осуществления средств, способов и приемов решения. Иными словами, в процессе проведения собственно учебной игры должны быть «вскрыты» в обобщенном виде общее отношение, правило, закон, которые дают описание определенного класса задач, ситуаций, процессов.

Удовлетворяющая названным требованиям игровая компьютерная среда представляет собой сложную многоуровневую систему, включающую исчерпывающий набор элементов «натурального» уровня предметности, позволяющий воспроизвести в наглядно-образном плане существенные свойства объекта изучаемой области знания.

При этом упомянутый исчерпывающий набор элементов должен допускать разные уровни обобщения, «естественные» для изучаемой области знания и позволяющие учащемуся строить модель объекта усвоения на том уровне обобщения, который ему доступен на данном этапе овладения материалом. Разумеется, должны предусматриваться (и использоваться) «естественные» для конструирования способы оперирования элементами модельного уровня предметной среды. Так, в нашем примере естественными элементами оперирования являются «захват» квадратика или другого «кирпичика», перемещение его в нужное положение (включая, если нужно, поворот) и закрепление его в нужном положении. Эти элементы оперирования обеспечиваются системой команд исполнителя (обучаемого), которая сообщается в начале игры в виде наглядного инструктажа (показа).

Следует также предусмотреть возможность опосредованного воздействия на «натуральные» элементы предметной среды и способы их взаимодействия. Например, предусматривается возможность числового задания параметров. Так, в нашем случае (во второй игре) обучаемый мог по своему желанию выбрать число треугольников и квадратов для заполнения фигуры. Предусматривается также перемещение элементов, построение таблиц, формул и т. д.

Наконец, следует предусмотреть простые способы синтеза псоледовательностей элементарных операций с натуральными и модельными элементами среды в более сложные команды, выполняющие целостные преобра-

зования (выработка собственного языка), а также управление элементами среды. Так, в пятой игре пакета «Геометрическая комбинаторика» рассматриваются задачи о «паркетах». Эта тема очень интересна и включает трудные и легкие геометрические задачи. Речь идет о том, чтобы заполнить большой кусок плоскости заданными «кирпичиками» (или «паркетинками»). Ребенок может, составив из двух или большего числа «паркетинок» некоторый комплексный элемент заполнения плоскости, либо объявить этот комплексный элемент новым составным «кирпичиком» и далее повторять его, либо же, составив конфигурацию из двух-трех «паркетинок», регламентировать продолжение покрытия по тому же закону или предусмотреть иные способы укрупненного построения из тех или иных блоков.

Заметим в заключение, что с учетом разного уровня подготовленности, склонностей учащихся должны быть предусмотрены простые способы перехода от одного уровня предметной деятельности к другому и возможность совмещения работы с элементами натурального и модельного уровня предметности.

Развивающими компьютерными играми учащийся подводится к осмыслению логических выводов о возможности или невозможности осуществления задуманной (или требуемой) конструкции. Рассуждения здесь принципиально различны при обосновании возможности или невозможности сконструировать ту или иную фигуру. В первом случае (показ возможности построения) осуществляется синтез. Ученик (скажем, в третьей игре) прикладывает «кирпичики» один к другому, постепенно видоизменяя конструкцию таким образом, чтобы получить требуемую конфигурацию. Как правило, возможность построения показывается именно его осуществлением с использованием частных приемов и комбинаторных операций (типа «подбор элемента к целому», «включение — исключение», «повтор элементов в аналогичных расположениях» и т. п.). Вообще, за исключением некоторых математических теорий, дающих общие формулы и методы решения задач строго очерченного класса, возможность решения задачи, как правило, устанавливается его непосредственным осуществлением (в полном или схематичном виде). Напротив, обоснование невозможности решения представляет собой более сложную логико-познавательную проблему и осуществляется либо полным перебором, либо применением некоторого инварианта.

Мы уже отмечали в своей работе [2], что простенькая логическая игра-головоломка «Волк, коза и капуста» дает при надлежащем оформлении повод для весьма содержательного обсуждения, представляющего интерес даже для старшеклассников. Эта деятельность — в форме диспута, коллективного обсуждения и решения — осуществляется без компьютера, сверх содержащейся в нем программной разработки. В этом обсуждении может принимать участие группа учащихся, учитель. Но повод к этой (по существу исследовательской) работе дает компьютерная игра. Подобные поводы имеются и в других развивающих компьютерных играх и, в частности, в пакете «Геометрическая комбинаторика». Так, в четвертой игре, при составлении различных прямоугольников из нескольких «уголков», мы приводили «рецепт» для определения того, можно ли заданный прямоугольник сложить из нескольких «уголков». Это решение математически очень интересно, его доказательство непросто и представляет собой хорошую задачу олимпиадного типа. И, конечно, компьютер здесь уже больше не нужен: идет творческий процесс, интеллектуальный поиск. Имеются такие нестандартные задачи олимпиадного типа и в других развивающих играх. По существу, именно в выходе на такие новые, творческие подходы и заключается особенность развивающих компьютерных игр. Конечно, глубина и сложность задач, возникающих при применении развивающих компьютерных игр, может (и должна) быть различной: от более простых, доступных менее подготовленным учащимся, до сложных проблем.

Весьма существенной чертой развивающих компьютерных игр является возможность эффективно обучать де-

тей с разным уровнем интеллектуального развития. Дети могут освоить обобщенный способ решения геометрических (и других) задач. Диагностика повышения уровня интеллектуального развития, который достигается с применением компьютерных развивающих игр, представляет собой важную психологическую проблему.

Итак, существует принципиальное различие между подходом к компьютерным играм как к развлекательной деятельности и подходом к проектированию компьютерных игр учебного назначения. В последнем случае познавательный мотив доминирует над сверхценным отношением к собственно программированию («хакерство») и поиску острых ощущений в процессе компьютерной игры (азарт, стремление к легкому времяпрепровождению, и т. п.). Демонстративность, зрелищность, характерные для чисто развлекательных игр, отступают на второй план в компьютерных развивающих играх, хотя, разумеется, дизайну и красивому привлекательному оформлению и здесь придается важное значение.

Развивающий эффект компьютерных игр зависит от возраста и индивидуально-психологических особенностей детей. Определение этих зависимостей требует специально организованной экспериментальной апробации создаваемых образцов. Наши данные дают основание полагать, что компьютеры крайне необходимы при осуществлении учащимися операций двух учебных действий: при прослеживании происхождения знаний и при моделировании выделяемого исходного отношения, конституирующего эти знания. Уже давно и многократно были описаны внешние особенности первого учебного действия (это происходит при изложении особенностей знаменитого сократовского, или эвристического, метода обучения). Однако долго не было технических средств, позволяющих учащимся «своими руками» предметно (а не в словесном плане) реконструировать внутреннее содержание процесса происхождения тех или иных понятий. На наш взгляд, этими средствами сейчас могут стать компьютеры, правильно «вписанные» в целостную учебную деятельность.

Отдельные моменты учебного моделирования также хорошо известны и используются при построении различных графических схем, но в них выражаются внешние особенности усваиваемых знаний. Их внутреннее содержание может быть смоделировано именно учебными компьютерами (само их функционирование может стать моделью учебных действий).

Таким образом, одна из главных проблем компьютеризации обучения, на разработке которой следует сосредоточить исследовательские и конструкторские усилия, связана с определением тех операций в учебных действиях, которые наиболее эффективно могут осуществляться с помощью компьютера. Результаты таких разработок могут стать основой достаточно развернутой теории компьютеризации обучения и проектирования компьютерных игр учебного назначения.

Литература

1. *Болтянский В. Г., Рубцов В. В.* Вопросы компьютеризации школьного обучения // Вопросы психологии. 1985. № 6. С. 177—178.

2. *Болтянский В. Г., Рубцов В. В.* Современные проблемы компьютеризации обучения // Проблемы компьютеризации и статистической обработки данных: Сборник трудов. М.: ВНИИСИ, 1989. С. 30—46.

3. *Давыдов В. В., Рубцов В. В.* Тенденции информатизации советского образования // Советская педагогика. 1990. № 2.

4. *Нежнов П. Г., Улановская И. М.* Компьютерная игра как фактор психического развития детей разного возраста // Человек — Творчество — Компьютер: Тезисы докл. к 8-му Международному конгрессу по логике, философии и методологии науки. Ч. 1. М., 1987. С. 142—144.

5. *Рубцов В. В., Каптелинин В. Н., Львовский В. А., Невуева Л. Ю., Улановская И. М.* Логико-психологические основы использования компьютерных учебных средств в процессе обучения (психологическая концепция) // Информатика и образование. 1989. № 3, 4.

Владельцам КУВТ-86

Творческий кооператив «Альтернатива»

объявляет о своем преобразовании в государственное предприятие при Институте информатики и вычислительной техники АПН СССР (НИИ и ВТ). Предприятие сохраняет название своего предшественника и является его правопреемником.

Предлагаем всем владельцам КУВТ-86 приобрести:

● пакет учебных программ по предмету «Основы информатики и вычислительной техники» с комплектом технической и методической документации. Пакет состоит из 8 дисков. Описание пакета и отдельных его частей см.: Информатика и образование. 1989. № 3,6; 1989. № 5; 1990. № 3.

Оплата по перечислению. Цены минимальные (для среднего учебного заведения — 1000 руб., всех учебных заведений района — 3000 руб., для города — 5000 руб., для области или края — 10 000 руб.). Это в 3—7 раз меньше цен на аналогичные услуги кооперативов, центров НТТМ и др.;

● систему программирования на языке Пролог для КУВТ-86, альтернативную всем ранее известным.

Цены соответственно 500, 1500, 2500, 5000 руб.;

● систему программирования на языке Паскаль для КУВТ-86.

Цены соответственно 500, 1500, 2500, 5000 руб.;

● систему программирования на языке Си для КУВТ-86.

Цены соответственно 200, 500, 1000, 2000 руб.;

● систему программирования на языке Форт для КУВТ-86.

Цены соответственно 150, 400, 800, 1500 руб.;

● автоматизированную систему Author для создания учебных демонстрационных и игровых программ в кодах БК-0010 — авторскую систему для КУВТ-86. Эта система позволяет пользователю, не знающему языков программирования, создавать эффективные и красочные программы в кодах БК-0010.

Цены соответственно 700, 2100, 3500, 7000 руб.

Предлагаем:

● укомплектовать наш пакет при его поставке манипуляторами типа «джойстик» и обеспечить их совместимость с программами пакета. Стоимость комплекта из 12 джойстиков 1000 руб. перечислением, что в 2 раза ниже рыночных цен на них;

● заключить договор об авторском сопровождении нашего пакета, что обеспечит его постоянное обновление и улучшение разработчиком и льготный доступ к новейшим разработкам. Стоимость услуг не более 50 % от стоимости первоначальной поставки.

Гарантируем:

● высокое качество программного обеспечения. Все программы созданы на языке ассемблера БК-0010;

● работоспособность всех наших программ, пересылаемость их по сети связи и возможность для них ввода-вывода на дисковод в классах КУВТ-86 всех типов и модификаций;

● поставку программ по почте на наших дисках в комплекте с подробной технической и методической документацией (на дисках);

● удобство и наглядность наших программ в работе;

● конкурентоспособность нашей продукции с программами других разработчиков (система «Рига» и др.).

Предлагаем всем разработчикам для классов КУВТ-86 сотрудничество в деле создания обучающих программных пакетов. Разработчикам, имеющим серьезные методические идеи, мы готовы передать на договорных условиях наши инструментальные средства для КУВТ-86 с их описаниями.

Наш адрес: 656099, г. Барнаул, пр. Социалистический, д. 60, каб. 15, предприятие «АЛЬТЕРНАТИВА».

ЧТО МОЖЕТ ЭВМ?

И ввод, и вывод

Японская фирма «Чори» начала выпуск оригинального устройства, которое можно считать специализированным компьютером. Оно невелико (примерно 4×4×15 см); с одной стороны установлена оптическая считывающая головка, с другой — игольчатая печатающая головка наподобие тех, что стоят в обычных матричных принтерах. В середине — электроника.

Если «оптической» стороной провести по листу бумаги, имеющееся на нем изображение будет считано и запомнено. Теперь можно провести по чистому листу печатающей стороной — изображение будет напечатано. А если подключить это устройство к карманному калькулятору (например, такому, как «Casio SF-9000» с памятью 64К байт), оно будет работать как мини-принтер.

Спасти и сохранить

За рубежом ПЭВМ ценятся обычно намного ниже, чем «заряжаемая» в них информация, поэтому проблема защиты сводится обычно к защите данных. Используемые для этого приемы разнообразны. Наиболее очевидный — пароль, без знания которого нельзя воспользоваться ПЭВМ или войти в систему коллективного доступа.

Однако почти на любой ПЭВМ можно загрузить ОС со своей дискеты, что дает возможность «обойти» пароль, установленный хозяином на жестком диске. На такой случай предусмотрено шифрование данных, не позволяющее прочесть их посторонним.

Прочесть нельзя — уничтожить можно! Чтобы затруднить такое вмешательство, используются специальные программы, делающие «невидимыми» (не печатающимися на экране при выводе директория) наиболее ценные файлы. Можно также зашифровать директорий.

Но самые радикальные средства защиты — «аппаратные», например выпускаемый фирмой Ergotron (США) «намордник», позволяющий запереть компьютер на замок так, что исключается доступ к его внутренностям и дисководам. Следующие шаги в этом направлении (выводящие из области защиты одной лишь информации) — прикрепляемый к столу ящик, в который можно запереть компьютер, клавиатуру, соединительные шнуры (фирма EMJ, выпускающая его, в течение трех лет после покупки выплачивает 5000 долл. за компьютер, украденный из этого ящика), и специальная плита, позволяющая прикрепить к столу без повреждения его поверхности ПЭВМ или принтер.

Последние средства особенно актуальны для СССР.

Информационный бум в Западной Европе

Западноевропейский общий рынок бизнесмены сравнивают с игровым полем, в разных углах которого — разные и постоянно меняющиеся правила игры. Несмотря на быструю интеграцию, национальная специфика в самом широком смысле — от законодательства до привычек потребителей — остается значительной и сильно осложняет освоение рынков. Однако «запираться» в своих границах — значит добровольно отдать инициативу конкурентам.

В результате возникают новые информационные потребности по анализу состояния отраслей, профилей предприятий, международному и национальному законодательству и стандартам и т. п. Как выяснилось, пресса, в том числе и специализированная, не может их удовлетворить. Тут-то и предстали в новом свете компьютерные банки данных. Их преимущество — доступность в любое время из любой точки Западной (увы) Европы; полнота сведений, сравнимая с полнотой, предоставляемой подшивками нескольких специальных изданий за несколько лет; возможность легкого выбора конкретных данных.

Эти банки делятся на три основные группы: фактографические (выдают краткие справки по техническим данным, балансам, ценам, статистике и т. п. в текстовом или числовом виде), текстовые (содержат полные тексты документов, например законов, экономических отчетов), рекомендательные (дают указания для дальнейшего поиска информации, ссылки на источники, перечни литературы). В частности, банк данных «Брокергайд» собирает и выдает информацию о более чем 6000 поставщиков; «Келекс» содержит правовые документы и обслуживает, в частности, депутатов Европарламента; в «Дайэнгайде» перечислено более 800 банков и баз данных; «Эбас» информирует о научных и научно-технических проектах.

Несмотря на удобства банков данных, бизнесмены испытывают затруднения в их использовании. Необходимость изучать язык запросов, четко представлять себе структуру информационной системы, уметь оптимально формулировать свои потребности, чтобы не тратить лишнее время (т. е. деньги!) на информационный поиск, отпугивает многих. Это на руку консультантам-брокерам, специализирующимся на работе с банками данных.

Сегодня, пока еще не спала волна интереса западных бизнесменов к советскому рынку, пока наше экономическое законодательство представляет собой винегрет из ежегодно меняющихся законов и еще более «подвижных» местных и ведомственных постановлений, пока способ ведения дел в СССР непонятен большинству иностранцев, банк данных с соответствующей информацией имел бы большую популярность. А если бы он имел простой (с точки зрения предпринимателей) интерфейс, соответствующая программа могла бы пользоваться спросом на внешнем рынке.

Намек понятен?

В. КАЗИЕВ
Кабардино-Балкарский государственный университет

Математическое моделирование и вычислительный эксперимент

Приводимые здесь некоторые типы моделей и примеры моделирования различных систем помогают воспитывать, как показывает опыт автора, способность к творческому использованию ЭВМ и методов информатики, умение строить и реализовывать на ЭВМ модели различных реальных систем, особенно плохо структурированных и трудно формализуемых. Их использование способствует деловому, заинтересованному общению преподавателя и обучаемого, стимулирует научный поиск, позволяет совершенствоваться в программировании (в широком понимании — от постановки задачи до получения результатов на ЭВМ). Эти и аналогичные модели, программы и планы вычислительных экспериментов использовались на занятиях с учениками, со студентами младших курсов КБГУ, на курсах повышения квалификации учителей школ КБАССР, на внутривузовской и на зональной (Северокавказской) олимпиадах студентов.

Укажем основные направления применения этих и других моделей и моделирующих программ.

1. Ознакомление с моделью, с планом математического моделирования и вычислительного эксперимента, предмодельный анализ данных, состава и структуры моделей.

2. Проведение моделирования и вычислительного эксперимента, создание и реализация с помощью преподавателя различных деловых и игровых ситуаций, например с помощью идентификации модели.

3. Разработка аналогичных моделей, планов вычислительных экспериментов, а также усложнение и модификация предложенных преподавателем; реализация принципов сборочного моделирования.

4. Разработка аналогичных моделирующих программ и комплексов, а также их модификация и оптимизация, реализация принципов сборочного программирования.

5. Анализ результатов эксперимента и установление некоторых причинно-следственных связей, анализ адекватности модели.

6. Моделирование компьютерных технологий решения задач и автоформализации профессиональных знаний, например, путем выделения в группе обучаемых постановщиков задач (конечных пользователей), «модельеров» (системных аналитиков), программистов и экспертов.

7. Проведение под руководством преподавателя теоретического обоснования результатов вычислительных экспериментов.

Модель 1. Однородная популяция. На ее примере продемонстрируем основные технологические этапы моделирования и вычислительного эксперимента и обучения им.

Содержательная постановка задачи.

Основная единица измерения динамики популяции (группы организмов одного вида) — численность или плотность на единицу площади (пространства), занимаемого популяцией. Изменение численности или плотности популяции определяется процессами рождения и гибели индивидов популяции. Взаимодействие популяции с окружающей средой описывается обобщенными коэффициентами рождаемости и смертности. Необходимо разработать модель (алгоритм, программу) изменений популяции и провести вычислительный эксперимент по ее исследованию.

Формулировка гипотез относительно поведения популяции, **предмодельный анализ и построение математической модели.** Пусть $x(t)$ — численность или плотность популяции в момент времени t. Через промежуток времени Δt численность равна $x(t+\Delta t)$. Относительный прирост $\delta(x)=[x(t+\Delta t)-x(t)]/\Delta t$. Примем следующую гипотезу: относительный прирост прямо пропорционален численности, т. е. $\delta(x)=kx(t)$, где k — коэффициент пропорциональности. Считая, что $\Delta t \to 0$, получим следующий закон изменения численности: $x'(t)=kx(t)$. Эта простейшая модель динамики популяции называется мальтусовской (по имени ученого, исследовавшего ее), или экспоненциальной, моделью, так как решение этого уравнения — экспонента $x(t)=x(0)\exp(kt)$; $x(0)$ — начальное значение численности (плотности). Эта простая модель (гипотеза) не отражает того факта, что ресурсы среды обитания популяции (пища, свет, вода и т. д.) ограничены, вследствие чего имеется верхняя граница численности, к которой стремится популяция и при приближении к которой возникает недостаток ресурсов. Недостаток ресурсов подавляет размножение — увеличивается смертность.

Коэффициент пропорциональности k (называемый коэффициентом прироста) может быть записан в виде $k=p-c$, где p — коэффициент рождаемости, а c — коэффициент смертности. Эти коэффициенты относительные величины, $0 \leqslant p,\ c \leqslant 1$. Модель Мальтуса примет вид $x'(t)=(p-c)x(t)$. Недостаток этой модели — в игнорировании того факта, что коэффициент смертности зависит от численности (плотности) $x(t)$. Учитывая это, уточним модель Мальтуса — будем считать, что $c=c(x)$. Наиболее простой вид такой зависимости задается линейной функцией $c=e+bx$. Подставляя c и обозначая $a=p-e$, получим $x'(t)=[a-bx(t)]x(t)$. Эта модель называется моделью Ферхюльста—Пирла, или логистической, так как решение этого уравнения имеет вид логистической функции:

$$x(t)=x(0)\exp(at)/[1+bx(0)\times \times(\exp(at)-1)/a].$$

Модель Ферхюльста — Пирла более точно отражает поведение популяции, ее взаимодействие с окружающей средой обитания.

В рамках этих моделей игнорируются половые, возрастные и генетические различия индивидов, задержка во времени реакций среды (популяции); a, b считаются постоянными. Коэффициенты a, b в модели Ферхюльста — Пирла называются соответственно коэффициентами роста (скорости роста) и сопротивления среды (скорости гибели).

Алгоритм моделирования. Заметим вначале, что при разработке любых алгоритмов обучаемым рекомендуются следующие способы представления: словесный — как наиболее простая и естественная форма; школьный алгоритмический язык, дополненный командами ввода-вывода,— как средство поддержки школьного учебника; блок-схемы — как простое и привычное для обучаемых графическое средство; структурограммы (схемы Насси — Шнайдермана; см., например, [1]) — как удобное и наглядное средство поддержки нисходящей технологии; Р-схемы (см., например, [2]) — как технологическая инструментальная система.

При составлении алгоритма моделирования будем опираться на дискретную модель Ферхюльста — Пирла, которая получается заменой $x'(t)$ на $\delta(x)$. При этом берем $\Delta t=1$ (сутки, ч, мин и т. д.). Дискретная модель имеет вид:

$$x_{i+1}=x_i+ax_i-bx_i^2,\ x_0=c,\ i=0,\ 1,\ldots,\ n, \quad (1)$$

где n — предельное время моделирования.

Полезно рассматривать два режима моделирования: коэффициенты a, b известны; коэффициенты a, b неизвестны. Во втором случае, задавая дополнительно два значения $x_1=\alpha$, $x_2=\beta$, из соотношения (1) получим систему двух алгебраических уравнений относительно a и b ($i=1, 2$), из которой найдем неизвестные a, b. Работа в этом режиме позволяет отрабатывать такую важную операцию компьютерного моделирования, как идентификация параметров.

Алгоритм моделирования состоит из следующих этапов (приводим для краткости только словесное описание алгоритма):

1. Выбор режима моделирования.

2. Задание входных данных для моделирования: c, n, a и b (или α и β), единицу измерения времени, тип измерения популяции (0 — численность, 1 — плотность), способ вывода результатов (0 — экран, 1 — принтер).

3. Прогноз численности (плотности) популяции с момента времени $i=1$ до момента $i=n$ по формуле (1), реализуемый простым циклом.

4. Поиск стационарного состояния популяции, т. е. момента времени i, начиная с которого значение x_i, $i\geqslant j$ постоянно или мало изменяется, например $|x_i-x_{i+1}|<\varepsilon$, где ε достаточно мало.

5. Поиск момента, когда популяция вырождается, вымирает.

6. Выдача, по желанию пользователя, графика $x=x(i)$ или эскиза этого графика.

Программа моделирования. В данном случае этап программирования достаточно прост и не рассматривается.

План вычислительных экспериментов. Так как моделирующая программа представляет материально реализованную систему «Однородная популяция», то, замещая эту систему программой, можно с ее помощью получить новую информацию, «проиграть» поведение этой системы в различных условиях, провести вычислительный эксперимент. В данной задаче может быть реализован, например, следующий план вычислительных экспериментов по программе.

Определить, как влияют коэффициенты роста и сопротивления среды на динамику популяции, например найти с помощью ЭВМ некоторые границы или значения a, b, для которых популяция растет или убывает при одной и той же начальной численности.

Определить, задавая различные $0<a$, $b<1$, для каких a, b достигается стационарное состояние (саморегуляция системы) и как зависит скорость его достижения от величин a, b при одной и той же начальной численности.

Величина $v=a/b$ характеризует так называемую емкость среды обитания. Оценить емкость среды для различных популяций (различных a, b, c). Проанализировать v для тех a, b, при которых популяция вымирает.

Определить и проанализировать такие значения a, b, которые слабо влияют на динамику популяции, например найти такое a (b), при котором популяция качественно ведет себя одинаково независимо от значения b (a).

Найти значения a, b, при которых наступают периодические колебания численности (плотности) популяции.

Определить, график какой функции напоминает график зависимости $x=x(i)$, $i=0, 1,..., n$ (или какие графики напоминает $x=x(i)$ на отдельных участках изменения i).

Промоделировать несколько популяций с неизвестными a, b.

Дальнейшее уточнение и усложнение модели: $a=a(i)$, $b=b(i)$, $i=0, 1,..., n$.

Далее следуют несколько аналогичных задач для самостоятельного решения.

Модель 2. Биологический культиватор. Рассмотрим устройство, в котором происходят размножение бактерий (клеток) со скоростью a, их гибель со скоростью b, а также приток извне со скоростью s. Изменение концентрации бактерий в таком культиваторе описывается соотношением

$x_{i+1}=x_i+s+ax_i-bx_i^2$, $x_0=c$, $i=0, 1,..., n$.

Для простоты здесь и ниже рассматриваются дискретные варианты моделей. План вычислительных экспериментов и усложнение модели — как в модели 1. Можно добавить выяснение влияния скорости притока s на скорость достижения стационарного состояния.

Модель 3. Популяция, дающая урожай. Из популяции в каждый момент

времени изымается часть (пример — популяция промысловых рыб). Динамика популяции описывается соотношением $x_{i+1}=x_i+ax_i-bx_i^2-kx_i$, $x_0=c$, $i=0,1,...,n$, где k — задаваемый коэффициент сбора урожая (скорость изъятия особей). План экспериментов и усложнение модели — как в модели 1. Можно добавить выяснение границ (значений) $k>0$, при которых численность не убывает (не ниже некоторого заданного планового уровня x^*).

Модель 4. Популяция с заданным законом лимитирования. Динамика популяции описывается соотношением $x_{i+1}=x_i+ax_i-b(x_i)x_i$, $x_0=c$, $i=0,1,....,n$, где $b=b(x)$ — задаваемая по желанию пользователя функция лимитирования, например:

$b(x)=\alpha x$, $b(x)=\alpha x/(1+\alpha x)$, $b(x)=\alpha(1-\exp(-\beta x))$,

$b(x)=\alpha x^{\beta}$, $b(x)=\alpha\ln(1+\beta x)$, $b(x)=\alpha/(\beta-x)-\alpha/\beta$.

План экспериментов и усложнение модели — как в модели 1.

Завершая на этом описание технологии моделирования и вычислительного эксперимента и обучения им на примере модели 1, приведем далее ряд специально подобранных моделей и планов экспериментов из различных областей, ибо только содержательные задачи и современные математические модели позволят получить выгоду от компьютеризации общества.

Модель 5. Хищник — жертва. Имеются популяции двух видов, один из которых питается другим. Поведение этой системы описывается соотношениями

$$\begin{cases} x_{i+1}=x_i+a_1x_i-b_1x_i^2-q_1x_iy_i, & x_0=c_1, \\ y_{i+1}=y_i-a_2y_i+b_2y_i^2-q_2x_iy_i, & y_0=c_2, \end{cases}$$

где x_i — численность (плотность) жертв, y_i — хищников, a_1, b_1 (a_2, b_2) — коэффициенты рождаемости и смертности жертв (хищников), q_1 — коэффициент защиты жертв, q_2 — коэффициент прожорливости хищников.

Примерный план вычислительных экспериментов:

выяснить, как связано время уменьшения численности жертв в m раз со временем увеличения численности хищников в m раз;

выяснить характер зависимости периодических колебаний численности жертв (хищников) от a_i, b_i, c_i, q_i, $i=1, 2$;

выяснить, как развивается популяция жертв (хищников) в условиях нулевой или малой смертности (рождаемости) хищников (жертв);

определить, как изменяются среднеарифметические значения x_i, y_i в зависимости от изменения параметров системы;

выяснить, при каких соотношениях a_1/b_1, a_2/b_2 выживает первый (второй) вид;

исследовать графики динамики каждого вида, а также зависимость $y=y(x)$.

Усложнение модели: считать a_i, b_i, q_i, $i=1, 2$ изменяющимися во времени.

Модель 6. Хищники — жертвы. Имеются популяции n видов хищников и m видов их жертв. Эту систему можно задать таблицей (матрицей) A элементов a_{ij}, где $a_{ij}=1$, если i-й хищник питается j-й жертвой и $a_{ij}=0$ в противном случае. Названия видов и таблица A вводятся в диалоговом режиме.

План экспериментов:

по запросу пользователя (название жертвы) определять всех опасных для нее хищников;

программно определять группы хищников, питающихся одними и теми же жертвами (группировать и выдавать состав групп);

программно определять все непересекающиеся группы, т. е. группы, питающиеся только разными жертвами;

программно определять, есть ли в системе случаи «самоедства».

Усложнение модели: ввести таблицу предпочтений i-м хищником одного вида пищи по сравнению с другим видом.

Модель 7. Эпидемия болезни. В изолированном поселке с населением m человек возникла эпидемия болезни, распространение которой описывается соотношениями

$$\begin{cases} x_{i+1}=x_i-bx_iy_i, \quad y_{i+1}=y_i-cy_i+bx_iy_i \\ z_{i+1}=z_i+cy_i \\ x_0=a_0,\ y_0=b_0,\ z_0=c_0, \end{cases}$$

где x_i, y_i, z_i — число здоровых, больных (инфицированных) и невосприимчивых (переболевших) в момент времени $i=0, 1, ..., n$, b — частота контактов больных и здоровых, c — величина, обратная среднему времени выздоровле-

ния. Имеется набор лекарств L_j, $j=1, 2, ..., l$. Эффективность L_j характеризуется величиной $0<a_j<1$. Если в некоторый момент времени $y\leqslant 0$, моделирование завершается. Если при $i=k$ величина y_k станет больше y^* — задаваемого критического уровня больных, то, по желанию пользователя, осуществляется либо «изоляция больных» (последовательными умножениями b на задаваемый параметр $0<q<1$ находится такое b, для которого в промежутке от k до n не достигается y^*), либо «экстренная терапия» (подбирается наиболее эффективное лекарство L_p, $1\leqslant p\leqslant l$, для которого a_p максимально среди a_j, $1\leqslant j\leqslant l$, выдается его рекомендация и последовательными делениями c на a_p находится новое c, для которого в $[k, n]$ не достигается y^*.

План экспериментов:

выяснить характер изменений x_i, y_i, z_i в зависимости от различных b, c, a_0, b_0, c_0;

выяснить скорость выздоровления при «изоляции» больных;

выяснить скорость выздоровления при «экстренной терапии»;

выяснить скорость выздоровления при комбинации «изоляции» и «экстренной терапии»;

выдавать и анализировать графики динамики $x=x(i)$, $y=y(i)$, $z=z(i)$.

Усложнение модели: коэффициенты b, c, a_j зависят от времени; подключение модели (подпрограммы) выдачи рецепта, курса лечения.

Модель 8. Кассир. В кассе магазина n_i купюр и монет достоинством 100, 50, ..., 1 рубль, 50, 20, ..., 1 копейка ($i=1, 2, ..., 15$). Покупатель приобретает m_j штук товара вида j, $j=1, 2, ..., k$. Стоимость одной штуки j-го товара — s_j руб. Необходимо определить стоимость S всей покупки, сдачу, выдать сдачу имеющимися в кассе купюрами и монетами. Случай, когда от покупателя получена сумма $P<S$, не исключать.

Эксперименты состоят в проигрывании различных ситуаций с различными n_i, k, s_j, P.

Усложнение модели: в случае невозможности выдачи сдачи имеющимися в кассе деньгами выводить сообщение о размене некоторых купюр и вводить реакцию пользователя.

Модель 9. Производство. Закон функционирования некоторого производственного предприятия задается соотношением

$$x_{i+1}=x_i+ax_{i-j}-bx_i,\ x_0=c,\ i=0, 1, ..., n,$$

где x_i — количество производимой в момент времени i продукции, a — коэффициент, определяющий часть этой продукции, направляемой для нужд самого производства (например, в качестве сырья), j — промежуток времени от выделения этой части продукции до его использования при выпуске новой, b — задаваемый коэффициент, характеризует степень износа оборудования на предприятии, c — начальное количество продукции. Считать $a=0$ при $i<j$.

План экспериментов:

проиграть различные производственные ситуации, задавая различные a, b, c, n;

выяснить влияние различных j на количество выпускаемой продукции при одинаковых a, b, c, n;

проанализировать график динамики выпуска продукции.

Усложнение модели: a, b зависят от времени.

Модель 10. Классификатор. Имеются n объектов (ботанических, географических и т. п.). Каждый объект обладает m свойствами. Необходимо сгруппировать (классифицировать) объекты в небольшое число групп, объединяя в одну группу все похожие объекты. Через x_{ij} обозначим значение j-го свойства i-го объекта (например, $x_{ij}=1$, если объект i обладает свойством j, $x_{ij}=0$ в противном случае). Два объекта i, j считаются похожими и заносятся в группу l, $1\leqslant l\leqslant s$, если

$$d_{ij}=\sqrt{(x_{i1}-x_{j1})^2+(x_{i2}-x_{j2})^2+...+(x_{im}-x_{jm})^2}<r_l$$

где s, r_l задаются.

Эксперименты состоят в классификации различных объектов, например птиц, экономических районов, учащихся и т. д.

Усложнение модели: изобразить группы объектов графически, осуществить динамическое изображение этих групп.

Модель 11. Распространение техноло-

гии. Имеется некоторая компьютерная технология (технологическое новшество), распространение которой в некоторой области описывается соотношением $x_{i+1}=x_i+ax^*x_i-ax_i^2$, $x_0=c$, $i=0, 1, ..., n$, где a — величина, характеризующая скорость распространения этой технологии, x_i — уровень охвата области технологией в момент времени i, x^* — уровень максимального охвата области технологией (уровень насыщения).

План экспериментов:

проиграть различные ситуации, задавая различные a, c при постоянном x^*;

проиграть ситуации при постоянных a, c и различных x^*;

проанализировать графики распространения технологии.

Усложнение модели: при приближении к x^* выдавать сообщение о необходимости новой, лучшей технологии и переходить к ее моделированию (изменяя a); рассмотреть случай $a=a(i)$.

Модель 12. Прогноз урожая. В результате проведенных на полях экспериментов и анализа полученных данных выведены экспериментальные зависимости

$$y=A_i-B_i10^{a_i+b_ix_i},\ i=1, 2, ..., 5.$$

где y — урожайность сельхозкультуры (ц/га), x_1 — глубина обработки почвы (см), x_2 — густота высева (ц/га), x_3 — содержание влаги в почве (кг/га), x_4 — температура почвы (°С), x_5 — содержание минерального питания (ц/га). В таблице приведены данные по одному из полей.

i	A_i	B_i	a_i	b_i
1	190	1	2,038	—0,005
2	70	70	0	—0,53
3	120	120	0	—0,018
4	161	161	—0,05	0,01
5	160	151	0	—0,34

План экспериментов:

осуществить прогноз урожайности по различным значениям различных факторов x_i;

проанализировать динамику урожайности, например, графически — изображая графики $y=y(x_i)$, $i=1, 2, ..., 5$.

Усложнение модели: выдавать (с помощью некоторых подмоделей) по результатам эксперимента план мероприятий для достижения наибольшего или задаваемого урожая; присоединить подмодель поиска зависимостей по экспериментальным данным, например, используя метод наименьших квадратов (см. модель 13).

Модель 13. Автоматизированная система обработки экспериментальных данных. Используя сведения о методе наименьших квадратов [3, с. 68], свойства основных элементарных функций и свойства преобразований, сводящих эти функции к линейной $u=c+dv$, находить функцию (параметры a, b), которая более точно описывает заданную табличную функцию. Наиболее точной считать ту функцию, которой соответствует минимальный квадратный корень из суммы квадратов разностей значений найденной и табличной функций, деленный на число данных. Пример функций из «банка формул» (в скобках — линеаризующие замены):

$y=1/(a+bx)$ $(v=x, u=1/y)$;
$y=x/(a+bx)$ $(v=x, u=x/y)$;
$y=ab^x$ $(v=x, u=\lg y)$;
$y=ax^b$ $(v=\lg x, u=\lg y)$;
$y=a+b\lg x$ $(v=\lg x, u=y)$.

План экспериментов состоит в поиске различных зависимостей (законов) по экспериментальным данным.

Ряд аналогичных моделей содержится в [4]. Большинство приведенных и ряд других моделей реализованы программно в среде ОС RT-11 ДВК-3 (Бейсик, Паскаль) и на УКНЦ.

Литература

1. *Уолш Б.* Программирование на Бейсике. М.: Радио и связь, 1988.

2. *Абрамов В. А., Дубровин В. С.* Прикладное программное обеспечение. Программное обеспечение микро-ЭВМ: В 11 кн. Кн. 3.— М.: Высшая школа, 1987.

3. Основы информатики и вычислительной техники: Пробное учебное пособие / Под ред. А. П. Ершова, В. М. Монахова. Ч. 1. М.: Просвещение, 1985.

4. *Казиев В. М.* Программирование и организация диалога с ЭВМ на языке Бейсик: Методические указания. Нальчик: Изд. КБГУ, 1987.

Г. ЛЕБЕДЕВ

О новом учебнике информатики

Издан новый учебник (Основы информатики и вычислительной техники / А. Г. Кушниренко, Г. В. Лебедев, А. Р. Сворень. М.: Просвещение, 1990), тираж 2 млн. экземпляров. Еще 2,5 млн. планируется издать в 1991 г.

На многочисленных встречах с учителями и методистами мне часто приходится отвечать на вопросы не только о содержании учебника, но и о том, что лежит в его основе, какие цели мы ставили, что считаем главным в курсе информатики, как относимся к преподаванию на Бейсике и т. д. В этой статье я постарался ответить на наиболее часто задаваемые вопросы и, кроме того, максимально полно выразить позицию авторов.

Что представляет собой новый учебник и чем он отличается от предыдущего?

В учебнике 1988 г. нормально проработана была лишь первая глава «Алгоритмический язык». Главы «Устройство ЭВМ» и «Применения ЭВМ» представляли собой просто беллетристические и к тому же достаточно сырые тексты, которые, может быть, и можно было прочесть, но по которым совершенно невозможно было что-либо преподать. Мы это понимали, но тем не менее решились издать такой — лишь наполовину готовый — учебник, считая, что, пока идет эксперимент по первой главе (а это год), мы успеем доработать все остальное.

Сейчас работа над учебником завершена. Глава «Алгоритмический язык» существенно переработана с учетом многочисленных замечаний учителей и методистов, значительно увеличено количество задач и упражнений. В остальном же это новый учебник, совершенно непохожий на вариант 1988 г.

Учебник состоит из введения и трех глав: «Алгоритмический язык», «Устройство ЭВМ» и «Применения ЭВМ».

Введение (с. 3—24)

§ 1. **Информация.** Кроме «зажигательного» введения про роль информации, информатики и ЭВМ параграф содержит материал, связанный с двоичным кодированием и единицами измерения информации. Цели: 1) дать представление о понятии информации в информатике как последовательности битов, байтов или символов; 2) показать отличия от обыденной трактовки информации как смысла или значимости сообщения; 3) дать представление об обработке информации как о техническом преобразовании, производимом по строгим формальным правилам и не связанном с восприятием информации или ее осмыслением. Упражнения на кодирование и декодирование слов, чисел и выражений; преобразование текстов и чисел по строгим правилам.

§ 2. **ЭВМ.** Содержит краткую историю вычислительной техники и описание основных частей ЭВМ. Упражнения нацелены на формирование представлений о технических возможностях школьных ЭВМ (объем памяти, быстродействие и пр.).

§ 3. **Обработка информации на ЭВМ.** Основная цель — дать представление об алгоритмизации (программировании), о роли ЭВМ и языка программирования в этой деятельности. Параграф заканчивается небольшим эссе о научно-техническом развитии общества и об отделении информационного производства от материального, а также объяснением лозунга А. П. Ершова «Программирование — вторая грамотность». Среди упражнений — известные головоломки «Волк, коза и капуста», «Ханойская башня», нахождение фальшивой монеты и др., в которых решением является *описание соответствующего алгоритма*, а не конкретное число или формула.

Глава 1. Алгоритмический язык (с. 25—145)

Эта часть учебника претерпела наименьшие изменения по сравнению с предварительным вариантом (Основы информатики и вычислительной техники / А. П. Ершов, А. Г. Кушниренко, Г. В. Лебедев, А. Л. Семенов, А. Х. Шень. М.: Просвещение, 1988) и частично описана в статьях (Информатика и образование. 1989. № 2, 4, 5, 6).

Если быть совсем кратким, то изучение

основ алгоритмизации и школьного алгоритмического языка происходит в специально сконструированной, очень простой модельной обстановке (Робот на клетчатом поле). Условия задач, как и система команд Робота, строго формализованы и не обсуждаются (хотя некоторые из них, например радиационная разведка коридора, легко «обосновываются» самими учащимися). Глава нацелена на решение большого количества задач с постепенным вводом все новых алгоритмических конструкций и методов алгоритмизации. Фактически это некоторый уровень письма и счета, при котором не обсуждается, почему мы пишем одни слова, а не другие, складываем яблоки, а не что-нибудь еще. На этом этапе изучение алгоритмического языка внешне выступает как самоцель. Глава завершается § 16, посвященным методам составления циклических алгоритмов (использование рекуррентных соотношений и инвариантов, составление однопроходных и рекурсивных алгоритмов).

При изложении материала мы старались снять излишнюю математизацию и выделить алгоритмическую составляющую курса в «чистом виде». Исполнитель Робот и его команды («**вправо**», «**влево**», «**закрасить**», «**слева стена**» и пр.) предельно наглядны, мгновенно усваиваются даже в IV классе и почти не требуют пояснений. Задачи подобраны так, что учитель может предоставить ученикам возможность самостоятельно придумывать команды «n **раз**», «**пока**», «**если**» и др. и лишь показывать *форму записи* этих команд в алгоритмическом языке. При таком подходе изучение алгоритмического языка почти не занимает времени, а учащиеся могут все свои силы направить на решение задач и изучение методов алгоритмизации.

Глава 2. Устройство ЭВМ (с. 146—168)

Основная цель всей главы — демистификация ЭВМ, формирование общего представления о происходящих в ЭВМ процессах. Это общее представление формируется на четырех уровнях.

1. Уровень высоких и низких напряжений, вентилей, триггеров и электронных схем (§ 17). Основная цель — показать, как физические процессы и явления можно использовать для хранения и обработки информации. Среди упражнений — рисование различных вентильных схем, в том числе для одно- и двухразрядного сумматоров.

2. Уровень машинных команд (§ 18). При изложении используются 16 простейших команд ЭВМ «Электроника УКНЦ». Основная цель — показать, что такое машинная программа и как достигается автоматизм работы процессора.

3. ЭВМ как комплекс разных устройств, связанных через магистраль и работающих под управлением процессора (§ 19). (Пример упражнения: «Для подключения Робота к УКНЦ выделены два адреса — 65352 и 65354. Придумайте способ кодировки команд Робота и передаваемой им информации. Используя эту кодировку, переведите в машинную программу алгоритмы: А1, А31, А38, А40».) Кроме того, § 19 содержит описание работы с исполнителями «Монитор» и «Клавиатура» из алгоритмического языка (упражнения варьируются от составления алгоритма очистки экрана до реализации простейшего графического редактора).

4. ЭВМ в целом, как единый аппаратно-программный комплекс, т. е. с учетом программного обеспечения (§ 20). Основная цель — завершить изложение устройства и работы ЭВМ и показать, как это связано с работой на алгоритмическом языке. В параграфе излагаются понятия начальной загрузки, компиляции, интерпретации, операционной системы. В упражнения, в частности, входит составление простейших (10—15 строк на алгоритмическом языке) компиляторов.

Глава 3. Применения ЭВМ (с. 169—217)

Глава начинается двумя параграфами, продолжающими тему алгоритмического инструментария. В § 21 вводится понятие информационной модели исследуемого объекта или процесса, которое иллюстрируется простейшими моделями кинозала, транспортной сети, моделями геометрических объектов на плоскости, обстановки на поле Робота и пр. Среди упражнений — задачи на составление, изменение и использование различных информационных моделей (например, паспорта, камеры хранения, расписания уроков и др.). В § 22 рассматривается понятие *информационной модели исполнителя*, вводятся новая конструкция алгоритмического языка «**исп-кон**» (аналог пакета в Аде или модуля в Модуле-2) и новый вид величин — общие величины исполнителей. Пример упражнения: «Используя исполнителя Чертежник, запишите на алгоритмическом языке модель исполнителя Черепашка со следующими командами: ...»

§ 23—27 посвящены разным областям применения ЭВМ и построены по единой схеме: сначала описание тех или иных применений ЭВМ в данной области; потом одна-две учебные системы (информационные модели соответствую-

щих исполнителей), демонстрирующие простейшие методы представления и обработки информации в этой области; в конце — упражнения как на изменение и пополнение приведенных, так и на составление новых информационных моделей.

§ 23, например, посвящен информационным системам. В начале бегло описаны система продажи железнодорожных билетов ЭКСПРЕСС; система управления сборочным конвейером ВАЗа; информационно-учетная система междугородней телефонной связи; база данных предприятий. В качестве учебных информационных систем рассматриваются системы «Вагон» (продажа билетов в один 36-местный вагон, следующий без остановок) и «Телефонная книжка» (справочная система о владельцах телефонов на 999 номеров). Примеры упражнений: «В рамках исполнителя «Телефонная книжка» придумайте способ кодирования информации об адресах абонентов. Составьте алгоритм, который по адресу дома определяет количество телефонов, установленных в этом доме»; «Придумайте способ кодирования информации и опишите общие величины исполнителя «Автомобили», предназначенного для хранения информации об автомобилях (номер, ФИО владельца, марка автомашины и ее цвет). Составьте алгоритм, выводящий на экран список всех владельцев белых «Волг», в номерах которых есть буква М и цифра 7».

§ 24 посвящен обработке текстовой информации (и содержит фрагменты простейшей модели исполнителя «экранный редактор»); § 25 — научно-техническим расчетам и приближенным вычислениям (нахождение корня функции методом деления отрезка пополам, вычисление интеграла методом трапеций, определение площади фигуры методом Монте-Карло); § 26 — вычислительным экспериментам на ЭВМ (в основном методу дискретизации непрерывных процессов на примере падения тела с учетом сопротивления воздуха); § 27 — компьютерному проектированию и производству и, совсем кратко, методу конечных элементов.

Заключительный параграф учебника (§ 28) содержит беглое и достаточно эклектичное описание других областей применения ЭВМ (спутниковая навигация автомобиля, всемирные информационные сети, электронный магазин и электронные деньги, компьютерные игры и т. д.), а также описание ситуаций, когда использование ЭВМ может принести вред, затормозить научно-технический прогресс либо привести к незапланированным или нежелательным последствиям.

Каково место курса информатики в средней школе?

Я считаю, что информатика в школе — это общеобразовательный базовый курс, основная цель которого — развитие специфического (принято говорить алгоритмического) стиля мышления. У человеческого мышления есть разные стороны. Математика, например, развивает математический (чаще говорят логический) стиль мышления. Литература, по-видимому, включает какие-то аспекты сопереживания, этики и морали. Ряд школьных предметов вообще не связан с какой-либо специфической стороной мышления, а нацелен на изложение знаний в конкретной области, на развитие кругозора учащихся. Я считаю, что современная школа должна не столько наделять выпускников конкретными знаниями и навыками, готовыми ответами, сколько закладывать общую базу мышления и культуры, формировать всесторонне развитого человека. С этой точки зрения математика как курс, в рамках которого преимущественно формируется логический стиль мышления, и информатика как курс, специально нацеленный на развитие алгоритмического мышления, должны в обязательном порядке входить в общеобразовательные базовые курсы средней школы. Более того, основы алгоритмизации, как и основы логики, в идеале должны закладываться на ранних стадиях обучения, не позже 10—12 лет.

Чем обусловлен выбор школьного алгоритмического языка?

В принципе мы могли бы использовать почти любой современный язык программирования, ибо целью является не изучение конкретного языка, а развитие соответствующих сторон мышления, умение решать задачи. Мы использовали (с минимальными изменениями) алгоритмический язык, предложенный А. П. Ершовым (фактически это некоторый ру-

сифицированный современный вариант Алгола-60), просто потому, что у нас не было достаточных оснований его менять — мы смогли выразить на этом языке все, что хотели. У меня, например, было очень большое желание сменить математическую терминологию первых учебников (алгоритм, вспомогательный алгоритм, величина) на общепринятую (программа, подпрограмма, переменная или объект и т. д.), но даже для этого мы не нашли достаточных оснований. Ибо, повторю, цель совсем в другом — в умении мыслить соответствующими категориями и решать задачи.

Как Вы относитесь к Бейсику и Паскалю?

Если говорить о традиционном Бейсике, то, во-первых, он не позволяет выразить такое фундаментальное понятие, как вспомогательный алгоритм (подпрограмма) с параметрами. Поэтому Бейсик, как бы ни был он распространен или любим, абсолютно не пригоден для курса основ информатики, если только мы под этим понимаем основы мышления, а не возможность понажимать клавиши на произвольно взятом компьютере. Во-вторых, при использовании как Бейсика, так и Паскаля масса сил учащихся уходит на особенности синтаксиса (англоязычная лексика, номера строк в Бейсике, точки с запятой и порядок расположения описаний и процедур в Паскале и пр.). Как следствие падает алгоритмическая сложность решаемых задач. С этой точки зрения алгоритмический язык существенно выигрывает и у Бейсика, и у Паскаля.

Вопрос об особенностях синтаксиса, правда, имеет и обратную сторону. Дело в том, что при использовании Бейсика интерес к предмету у учащихся первое время выше, чем при использовании алгоритмического языка. Я это объясняю тем, что программа на Бейсике выглядит загадочно (как «настоящая»), какой бы элементарной ни была задача и как бы плохо или хорошо программа ни была написана. Поэтому учащиеся довольно быстро начинают чувствовать себя «посвященными», и это поддерживает интерес к предмету. Беда лишь в том, что этот интерес не связан с информатикой как таковой. Представьте, что в школьном курсе математики числа требуется записывать в римской системе счисления, а слова некоторым образом шифровать. Конечно, можно писать и так, но какое отношение это имеет к содержанию курса математики и развитию логического мышления, даже если это интереснее, чем доказательство теорем и решение задач? Примерно так же я отношусь и к Бейсику в курсе информатики.

Разделяете ли Вы понятия алгоритмизации и программирования, алгоритмического языка и языка программирования?

В самых первых учебниках информатики А. П. Ершова алгоритмический язык использовался как специальный язык для записи алгоритмов людьми. Считалось, что при выходе на ЭВМ алгоритм *обязательно требует* переписывания на одном из языков программирования. Я считаю, что такое различение алгоритмизации и программирования — одна из самых крупных ошибок первых учебников. Да и сам Андрей Петрович говорил, что в учебниках он имеет в виду программирование в некоем «узком смысле», в отличие от «широкого смысла», заложенного в лозунге «Программирование — вторая грамотность».

Надуманность противопоставления алгоритмического языка языкам программирования показали и немедленно появившиеся реализации алгоритмического языка на ЭВМ, которые фактически придали алгоритмическому языку статус одного из языков программирования и сняли проблему рутинного, чисто технического переписывания алгоритмов с алгоритмического языка на язык программирования.

Поэтому в новом учебнике понятия алгоритмизации и программирования не различаются и используются в одинаково «широком» смысле. Алгоритмический язык рассматривается как один из языков программирования, а алгоритмом называется программа, записанная на алгоритмическом языке.

Вам не кажется, что Робот ребятам быстро надоедает?

Материала, связанного с Роботом, в учебнике действительно много, и он может надоесть. Но я бы отметил здесь три момента.

1. Очень многое зависит от учителя. Если ученики ждут знакомства с ЭВМ, операционными системами и компиляторами, справочных сведений о том, какими командами вызывается то или иное действие, то Робот будет восприниматься как нечто «ненастоящее», не имеющее отношения к делу. Этот негативизм можно предотвратить, если с самого начала умение *думать и решать задачи* поставить выше технических знаний и навыков работы на ЭВМ.

2. Если учитель считает необходимым, он может разнообразить материал, включив кроме Робота и Чертежника других исполнителей, быть может придуманных самими учащимися.

3. При работе на алгоритмическом языке чем лучше решение, тем очевиднее кажется ответ. Поэтому у некоторых учащихся может возникать ощущение «игрушечности» задач («Что мы все Робота гоняем!»), а у других — при сравнении с чужими результатами — разочарование («Это ж так просто!»). Чтобы не потерять интерес, учитель должен индивидуально варьировать сложность задач в зависимости от уровня учеников.

Пусть, например, Робот стоит слева от вертикальной стены и требуется составить алгоритм, который заставит Робота обойти стену вокруг и закрасить все примыкающие к стене клетки. Значительная часть класса сделает такую задачу быстро. Пока остальные раздумывают или (в машинном варианте) вводят свои алгоритмы в ЭВМ, тех, кто задачу уже сделал, можно попросить изменить алгоритм применительно к более сложной ситуации. Например, если от исходной вертикальной стены кое-где влево и вправо могут отходить выступы размером в одну клетку или выступы неизвестной длины. При каждом таком усложнении будет происходить очередное расслоение класса. Учитель может заготовить десяток разных усложнений и предлагать те или иные варианты в зависимости от уровня и особенностей ученика, наличия или отсутствия ЭВМ и пр. Таким образом можно создать и напряженную рабочую обстановку для сильных учеников и предоставить самым слабым возможность *самостоятельно* справиться с исходной задачей.

Не кажется ли Вам преподавание информатики без компьютеров бессмысленным и похожим на обучение плаванию без бассейна?

Нет, не кажется. Научиться пользоваться компьютерами без компьютеров действительно нельзя. Но развивать алгоритмический стиль мышления можно независимо от того, есть в школе компьютеры или нет, и даже в принципе независимо от того, существуют ли компьютеры вообще. Компьютер для нас не цель, а лишь средство обучения, такое же, как, например, мел и доска. Другое дело, что использование компьютеров позволяет интенсифицировать обучение, создать дополнительные мотивационные стимулы для учащихся, т. е. сделать обучение более эффективным и интересным. Однако весь материал учебника можно изучать как в машинном, так и в безмашинном варианте.

Зачем всех учить программированию, ведь только единицы станут профессиональными программистами? Не лучше ли учить пользоваться готовыми системами?

Это общий вопрос о том, должно ли образование закладывать основы мышления и общей культуры или формировать конкретные навыки, которые понадобятся, скажем, рабочему у станка. Применительно к физике, например, вопрос можно переформулировать так: «Зачем всех учить основам электричества (сопротивление, ток, закон Ома), ведь только единицы станут профессиональными электриками; не лучше ли учить включать/выключать свет и выворачивать лампочки?» Лично я — за основы мышления и культуры. Думаю, что подход «это им в жизни не понадобится» является деструктивным: а) по-

тому что мы не можем предугадать, у кого как сложится жизнь и б) потому что при таком подходе даже умение читать и говорить перестанет быть общим элементом культуры, ибо есть профессии, где и это не надо.

Не подменяется ли у нас информатика программированием?

Беда в том, что на практике чаще учат конкретному языку программирования, а не программированию как таковому. Это все равно как если бы мы пытались научить человека решать математические задачи, изучая устройство ручки и бумаги. Я считаю, что здесь есть только один путь — это создание обстановки, в которой учащиеся будут *сами* решать разнообразные задачи все возрастающей сложности, требующие все более глубокого изучения основ алгоритмизации. Именно по этой причине в новом учебнике упражнениями снабжены все параграфы, кроме последнего, а общее число упражнений резко увеличено.

Что же касается призыва не подменять информатику программированием, то им, к сожалению, слишком часто прикрывают элементарное незнание предмета, помноженное на желание выглядеть специалистом в этой области. Ибо для того, чтобы рассуждать об алгоритмах типа перехода улицы или заварки кофе, говорить о «понятности» алгоритма исполнителю или изучать такое свойство информации как «своевременность», вообще ничего знать не надо. Это даже проще, чем перечислять операторы Бейсика, не говоря уже о том, чтобы учить думать и решать задачи, учить основам алгоритмизации, включая все стадии этого процесса.

Ваше отношение к учебнику под редакцией Каймина?

Качество учебников пусть оценивают учителя, методисты, эксперты. Я могу говорить только о разнице в подходах.

В учебнике под редакцией В. А. Каймина упор сделан на основы логики и язык Пролог. Информатика при этом фактически превращается в один из разделов математики и может способствовать повышению лишь математической, логической культуры учащихся. Пролог как раз и является попыткой задействовать эту культуру при практической работе на ЭВМ.

Мы же главным предназначением информатики считаем развитие алгоритмических аспектов мышления как необходимого условия разностороннего развития личности (независимо от того, на каких языках и как мы будем работать на ЭВМ). И дело здесь не в терминах (что именно есть информатика), а в том, что информатика сейчас — единственный такой предмет в школе.

Существует ли методическое пособие по учебнику?

Методическое пособие для учителей (А. В. Авербух и др.) издательство «Просвещение» планирует издать в 1991 г.

Какое программное обеспечение курса уже существует?

Учебная система программирования на основе школьного алгоритмического языка «Е-87» (Е-практикум) и Комплект Учебных Миров КуМир*, включающий в частности Робота и Чертежника, реализованы для КУВТ «Ямаха MSX-1», «Ямаха MSX-2», «Корвет», УКНЦ и даже для КУВТ-86 (87) на основе БК-0010. Существуют уже и национальные варианты этих систем, переведенные на языки некоторых союзных республик. Справки о том, кто, как и на каких условиях поставляет эти системы, можно получить по тел. 264-30-22 (предприятие «ИнфоМир»). Для КУВТ «Ямаха», например, версии этих систем, выпущенные ранее 1990 г., объявлены свободно копируемыми и распространяются без всяких условий.

В настоящее время завершается работа над новой версией системы «КуМир» (включая и новую систему программирования), которая будет работать на КУВТ «Ямаха», «Корвет», УКНЦ и на

Подробнее о КуМире см.: ИНФО. 1990. № 3.— *Примеч. ред.*

ПЭВМ, совместимых с IBM PC. В новой версии язык приведен в полное соответствие с учебником (включая многословные имена), максимальный размер программы существенно увеличен, предусмотрено подключение новых исполнителей. Мы предполагаем вместе с новой версией поставлять учебные базы данных и электронные таблицы, учебные системы «Вагон», «Телефонная книжка», «Экранный редактор» и др., написанные на школьном алгоритмическом языке, с тем чтобы учащиеся могли не только познакомиться с внешним видом, но и изучить методы представления и обработки информации в той или иной системе, изменить или дополнить эту систему и т. п. Мы рассчитываем также, что характеристики новой версии позволят создавать с ее помощью интерактивные учебные системы по другим школьным предметам.

Есть ли опыт преподавания по учебнику?

Преподавание по предыдущему учебнику (тираж 139000 экз.) ведется с 1988 г. Мы лично взаимодействовали с учителями и методистами примерно 30 московских и подмосковных школ. Многие изменения в новом учебнике появились именно в результате их замечаний и предложений.

Опыт показал, что учебник «нормально идет» как в безмашинном, так и в машинном (при наличии соответствующего программного обеспечения) варианте. Сложности возникают лишь при использовании учебника в машинном варианте, но без адекватного программного обеспечения. В этом случае приходится либо перекраивать материал, либо проходить значительную часть в безмашинном, по сути дела, варианте.

В ряде школ в порядке эксперимента велось преподавание не только в IX—X, но и в VII—VIII (по старой нумерации) классах. Опыт показал, что в VII—VIII классах учебник идет намного лучше, чем в IX—X (этот результат, впрочем, скорее всего, объясняется возрастными особенностями учеников и слабо связан с конкретным учебником или даже предметом).

На какой возраст рассчитан учебник?

Учебник издается для X—XI классов с учетом программы на 102 ч. Конечно, X—XI классы не очень удачный возраст для того, чтобы «гонять» Робота. В VIII классе этот материал идет во много раз лучше. В идеале же, я считаю, что основы алгоритмизации должны изучаться в V—VI классах, а более сложные элементы, например методы алгоритмизации и применения ЭВМ, — в VIII—IX классах. Соответствующие изменения в учебник мы могли бы внести почти мгновенно. Но здесь сложилась удивительная ситуация: за спуск курса информатики в VIII—IX классы выступают все — учителя, методисты, авторы учебников, люди со стороны, даже сотрудники ГКНО. Однако воз и ныне там. Поэтому единственное, что мы смогли сделать,— это локализовать «старшеклассный» материал, с тем чтобы учитель мог его пропустить при преподавании по учебнику в VIII—IX классах.

М. САПИР

Курс ОИВТ — каким мы его видим

1. Цели

Основная цель — подготовить школьника к жизни в информатизированном обществе. Для этого его нужно научить *технологии решения задач с помощью ЭВМ*.

Прослушав курс ОИВТ, учащийся должен уметь ответить на вопросы: какие задачи (классы задач) решаются с помощью ЭВМ, каковы основные этапы их решения, в чем суть информационных технологий?

2. Структура курса

Мы считаем, что в курсе информатики нужно обучать трем базовым технологиям: *составления моделей задач, составления алгоритмов решения задач, использования программного обеспечения ЭВМ*.

Технология составления моделей сознательно поставлена на первое место: это основная технология информатики.

В связи с этим мы делим курс на следующие основные разделы.

1. Знакомство с ЭВМ — первоначальное знакомство с базовым программным обеспечением или, иначе, компьютерная грамотность.

2. Первоначальные сведения о том, что значит «поставить задачу», что такое «модель задачи» и каковы этапы решения задач с ЭВМ (модель — алгоритм — вычислительный эксперимент).

3. Алгоритмизация — изучение технологии построения алгоритмов с одновременным углублением знаний о других базовых технологиях.

4. Дальнейшее знакомство с базовым программным обеспечением, включая изучение основ реального языка программирования и основных принципов строения ЭВМ.

3. Программное обеспечение курса и его назначение

Принцип, которым мы руководствовались при создании программного обеспечения, таков: программное средство должно обеспечивать достижение соответствующей методической цели и при этом быть как можно более простым в использовании. Двухлетний опыт использования этих педагогических программных средств на уроках информатики показывает, что этот принцип удалось выдержать. На освоение каждого из этих программных средств ученики затрачивают не более 15 мин. Восстановление навыков после пропуска двух-трех занятий — не более 5 мин.

Программное обеспечение курса реализовано в полном объеме на ЭВМ IBM PC, «Ямаха», «Корвет» и в 1989 г. удостоено Золотой медали ВДНХ СССР. Оно состоит из следующих программных педагогических средств.

3.1. Исполнитель «Чертежник»: простой исполнитель с минимальным набором действий. Назначение: обучение технологии организации действий в алгоритмах.

3.2. Исполнитель «Робот-манипулятор»: простой исполнитель, работающий на прямоугольном стеллаже с деталями. Может перекладывать деталь из ячейки стеллажа в грузовой отсек и обратно, перемещаться вдоль стеллажа и т. д. Назначение: обучение технологии организации данных (табличная форма организации данных и стек).

3.3. Исполнитель «Вычислитель» предназначен для обработки числовой информации. Назначение: отработка навыков составления алгоритмов при решении реальных вычислительных задач; обучение технологии проведения вычислительного эксперимента; обучение этапам решения задач на ЭВМ.

Имеется четыре режима работы с исполнителями. Первый — программный. Ученик может писать и отлаживать программы, управлять исполнителем непосредственно, оттрассировать свою программу (при трассировке исполнитель подробно комментирует свою работу). Во втором режиме ученик должен вручную оттрассировать учительскую про-

грамму, отвечая на вопросы исполнителя. Этот режим может использоваться для текущего контроля. При третьем режиме ученику предлагается восстановить программу учителя по результатам ее работы. Четвертый — позволяет учителю самому создавать демонстрационные программы, которые могут не только давать информацию ученику, но и задавать ему вопросы.

3.4. Учебное базовое программное обеспечение: информационно-поисковая система, электронная таблица, редактор текстов, графический редактор. Учебный пакет прикладных программ «Симплекс метод». Назначение: обучение технологии работы с базовым программным обеспечением.

3.5. Имитатор микро-ЭВМ «Кроха»: трехадресная ЭВМ с восемью ячейками памяти, двумя регистрами (регистр команд и сумматор), дисплеем. Все это — память, регистры и дисплей — помещается на экране ЭВМ, которая имитирует «Кроху». Это делает работу с «Крохой» очень наглядной. Назначение: обучение трем принципам фон Неймана, лежащим в основе конструирования ЭВМ (принцип двоичного кодирования, принцип хранимой программы и принцип программного управления); общее знакомство с основным алгоритмом работы процессора.

4. Алгоритмизация

Цель алгоритмизации — обучение основным способам организации действий и данных, а также применению алгоритмических конструкций при составлении алгоритмов решения разнообразных классов задач. Мы против использования жестко регламентированного алгоритмического языка (в частности, школьного алгоритмического языка). За воспитанной школьным алгоритмическим языком синтаксической правильностью алгоритмов может скрываться полное непонимание учеником сути алгоритмизации.

Мы считаем, что алгоритм в тетради ученика или на классной доске должен быть текстом, построенным по правилам естественного языка.

5. Исполнители алгоритмов

Мы являемся сторонниками активного использования исполнителей. Понятие исполнителя в информатике фундаментальное, поскольку любая программа является, по сути дела, имитацией на ЭВМ какого-либо исполнителя с ограниченным набором допустимых действий. Прежде чем решать на ЭВМ задачи некоторого класса, нужно сымитировать на ЭВМ соответствующего исполнителя.

Кроме того, исполнители, сымитированные на ЭВМ или другим способом, играют в курсе информатики роль технических средств обучения.

Для достижения различных методических целей нужны различные исполнители.

6. Логика при изучении информатики

Мы считаем, что изучение логики высказываний никак не связано с изучением алгоритмизации. Ни при изучении алгоритмических конструкций ни при изучении основных способов организации данных не нужно умение работать со сложными высказываниями и таблицами истинности. Логические основы строения ЭВМ слишком сложны, чтобы их можно было изучить за отведенное информатике учебное время. Язык Пролог (для его использования знания логики необходимы) в существующем виде слишком сложен для использования в школе (можно использовать лишь очень малую часть его возможностей). Кроме того, изучение непроцедурного Пролога не вписывается в освоение трех базовых технологий, указанных выше. Утверждение, что Пролог — основной язык общения человека с ЭВМ будущего, останется бездоказательным до тех пор, пока это будущее не наступит. Изучать Пролог впрок нам кажется бессмысленным. Скорее всего, к тому времени, как будет создана «ЭВМ будущего», будет придуман другой, более удобный и мощный язык общения с ЭВМ. Представляется совершенно неочевидным, что математическая логика будет являться его основой.

7. Информационные технологии

Изучение новых информационных технологий — одна из основных целей курса информатики. Важно выделить основные элементы этих технологий и обучать именно им. «Профессиональные» программные средства (базы данных, электронные таблицы и т. д.), по нашему мнению, совершенно непригодны для изучения на обычных (нефакультативных) уроках информатики. Нужны средства, обладающие основными функциями «настоящих» программных средств, но значительно более простые в общении.

8. Моделирование

Каждая наука, а значит, и каждый школьный предмет занимаются составлением информационных моделей реальных задач. Отличие между науками состоит в различии методов и средств построения моделей. В то же время построение моделей во всех науках имеет много общего: существует единая базовая технология составления информационных моделей, которая должна быть изучена отдельно. Для этого и служит предмет «Основы информатики и вычислительной техники».

9. Языки программирования

Мы считаем, что в курсе информатики необходимо изучение языка программирования, который мы рассматриваем как составную часть базового программного обеспечения ЭВМ. Следовательно, его знание — составная часть технологии использования программного обеспечения ЭВМ. Мы считаем, что не имеет особого значения, какой из процедурных языков программирования будет взят за основу. Важно научить школьников тому, как изучать любой язык программирования, поскольку изучение всех языков программирования основано на схожих принципах.

10. Раздел «Основы вычислительной техники»

Цель раздела «Основы вычислительной техники» — познакомить школьников с тремя принципами фон Неймана, на которых основывалось и пока основывается создание любой ЭВМ. Более глубокое изучение строения ЭВМ нецелесообразно ввиду отсутствия учебного времени. Физические основы ЭВМ, по-видимому, естественно изучать на уроках физики.

11. Сроки начала преподавания

«Основы информатики и вычислительной техники» должны изучаться в базовой школе, т. е. до или во время того, как конкретизации трех упомянутых технологий будут изучаться на других предметах.

12. Перспективы курса

«Основы информатики...» не могут быть растворены в других предметах, поскольку остальные предметы изучают лишь конкретизации указанных технологий. Нам кажется, что компьютерная грамотность (первоначальные навыки общения с ЭВМ) может быть выработана у школьников тогда же, когда закладываются основы обыкновенной грамотности,— в начальной школе. При этом, конечно, необязательно вводить особый школьный предмет. В VIII—IX классах нужно изучать информатику более глубоко, предмет «Основы информатики...» в этих классах необходим.

13. Структура программно-методического комплекса

Наш программно-методический комплекс состоит из трех частей: учебника, книги для учителя и программного обеспечения. Об учебнике подробно рассказывалось в наших статьях в «ИНФО». В 1989 г. он вышел тиражом 115 тыс. экземпляров. В этом году предполагается допечатка 200 тыс. экземпляров. В 1991 г. издательство «Просвещение» предполагает издать 300 тыс. экземпляров учебника. Книга для учителя содержит решения всех задач (более пятисот) из учебника, а также методику преподавания информатики (объем книги для учителя — свыше 1000 с.) Программное обеспечение описано выше.

14. Где можно освоить наш комплекс?

В Свердловске систематически проводятся семинары для учителей. Авторы учебника проводили и готовы проводить семинары в других городах.

15. Порядок приобретения

Для приобретения программно-методического комплекса надо обратиться к любому из членов авторского коллектива: А. Гейну, Е. Линецкому, М. Сапиру или В. Шолоховичу по адресу: 620219, Свердловск, просп. Космонавтов, д. 26, Свердловский государственный педагогический институт, лаборатория микропроцессорной техники. Телефон: 57-83-63.

М. СТЕПАНОВ
НИИ школ МНО РСФСР

Система индивидуальных заданий для практических занятий по информатике

Преподавателю информатики при проведении практических, факультативных и кружковых занятий необходима обширная и хорошо разработанная система заданий различного уровня сложности. Характерной особенностью информатики как школьного предмета является быстрое расслоение учащихся по степени заинтересованности и как следствие по уровню подготовленности. Преподавателю приходится одновременно работать и с учащимися, у которых надо постоянно поддерживать минимальный интерес к информатике, и с учащимися, познания которых по ряду вопросов превосходят знания самого преподавателя. Система заданий должна стать гибким инструментом, обеспечивающим индивидуальный подход к учащимся с различными интересами и уровнем подготовки.

В книге [1] разработана система заданий для проведения практических занятий на языке MSX-Бейсик (стандарт MSX-1 с ориентацией на базовый КУВТ «Ямаха»). Расширению этой системы по ряду направлений посвящена настоящая статья.

Задания, рассчитанные на учащихся с недостаточным интересом к информатике, как правило, должны предполагать получение определенного эффекта, обращенного к эстетическому чувству ученика. Одним из направлений, позволяющих возбудить дополнительный интерес к работе на компьютере, является использование цвета на ранних этапах обучения. При этом учащимся предоставляется возможность выполнить свою программу на компьютере учителя, оснащенном цветным дисплеем. (Предварительно учащийся выполняет свое задание, пользуясь монохромным дисплеем, на котором цвета заменены оттенками, а затем программа пересылается по сети на машину учителя.)

Примером простого задания, дающего хороший эстетический эффект, может послужить задача о построении и закраске областей, ограниченных семейством уменьшающихся разноцветных концентрических окружностей:

```
10 COLOR,1,1:SCREEN 2
20 FOR I=15 TO 1 STEP -1
30 CIRCLE(128,96),I*8,I
40 PAINT(128,96)
50 NEXT I
60 GOTO 60
```

Использование различных цветов при работе с текстами требует введения команды OPEN «GRP:» AS #1 (при этом нет необходимости в детальном разъяснении смысла этой команды, достаточно дать понятие о ее назначении и спо-

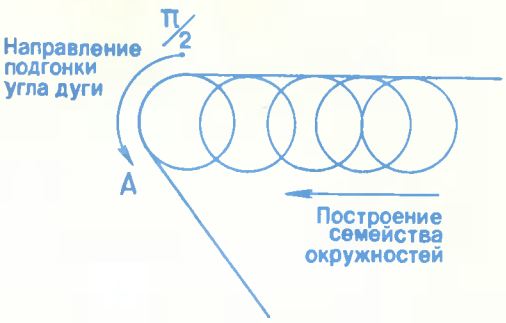
собе применения). Приведем примеры заданий для печати разноцветных слов и букв:

ЗАДАНИЕ	РЕШЕНИЕ
Напечатать слово по вертикали разноцветными буквами	10 COLOR,1,1:SCREEN 2 20 OPEN "GRP:" AS #1 30 PSET(0,50),1 40 COLOR 6:PRINT #1,"П" 50 COLOR 2:PRINT #1,"З" 60 COLOR 7:PRINT #1,"В" 70 COLOR 10:PRINT #1,"М" 80 GOTO 80
Напечатать слово по горизонтали разноцветными буквами	10 COLOR,1,1:SCREEN 2 20 OPEN "GRP:" AS #1 30 PSET(0,50),1 40 COLOR 6:PRINT #1,"П"; 50 COLOR 2:PRINT #1,"З"; 60 COLOR 7:PRINT #1,"В"; 70 COLOR 10:PRINT #1,"М" 80 GOTO 80
Напечатать разноцветные слова лесенкой	10 COLOR,1,1:SCREEN 2 20 OPEN "GRP:" AS #1 30 FOR I=2 TO 15 40 PSET(I*10,I*10),1 50 COLOR I 60 PRINT #1,"ПЭВМ" 70 NEXT I 80 GOTO 80

У определенной категории учеников устойчивый интерес вызывают программы, которые предназначены для построения различных рисунков в графическом режиме. Переход к иным программам нередко вызывает падение интереса к работе на ПЭВМ. В этом случае желательно мотивировать введение новых операторов как средства, предназначенного для построения тех или иных элементов рисунка.

Задание. При построении изображения необходимо провести дугу окружности, касающуюся двух прямых — горизонтальной и наклонной, так, чтобы она сопрягала эти прямые.

Р е ш е н и е. Выбрав радиус, мы можем определить ординату центра искомой окружности, исходя из того, что она касается горизонтальной прямой. Этой прямой касается семейство окружностей данного радиуса с данной ординатой центра (рис. 1). Для нахождения абсциссы используем построение семейства окружностей. Очередная окружность будет строиться только после нажатия на какую-либо клавишу. На



экране после ее построения будет печататься текущее значение абсциссы:

```
10 COLOR,1,1:SCREEN 2
20 OPEN "GRP:" AS #1
30 LINE(52,160)-(34,74)
40 LINE(30,70)-(100,70)
50 FOR X=80 TO 0 STEP -1
60 CIRCLE(X,77),7:REM ВЫБРАН РАДИУС 7
70 LINE(180,50)-STEP(30,10),1,BF
80 PRESET(180,50)
90 PRINT #1,X
100 F$=INPUT$(1)
110 NEXT X :REM РЕЗУЛЬТАТ : X = 42
```

Удаление выступающих концов отрезков, а также определение углов, задающих сопрягающую дугу окружности, можно частично рассчитать, а частично подогнать аналогичным способом. Например, подгонка дуги может быть произведена следующей программой.

```
10 COLOR,1,1:SCREEN 2
20 OPEN "GRP:" AS #1
30 LINE(52,160)-(35,78)
40 LINE(42,70)-(100,70)
50 FOR A=3.14/2 TO 2*3.14 STEP .05
60 CIRCLE(42,77),7,,3.14/2,A:REM УГОЛ
                  π/2 ВЫЧИСЛЯЕТСЯ ЗАРАНЕЕ
70 LINE(180,50)-STEP(40,10),1,BF
80 PRESET(180,50)
90 PRINT #1,A
100 F$=INPUT$(1)
110 NEXT A :REM РЕЗУЛЬТАТ : A = 3.17
```

Примененный выше метод визуальной подгонки дуг, отрезков и т. д. может использоваться при построении довольно сложных изображений.

Задание. Постройте на экране эмблему общества «Спартак».

Р е ш е н и е. Подгонку концов отрезков так, чтобы они лежали на границе буквы С, можно произвести визуально.

```
10 COLOR 6,15,15:SCREEN 2
20 OPEN "GRP:" AS #1
30 LINE(128,10)-(0,80)
40 LINE(128,10)-(255,80)
50 LINE(128,150)-(255,80)
60 LINE(128,150)-(0,80)
70 LINE(128,20)-(20,80)
80 LINE(128,20)-(235,80)
90 LINE(128,140)-(235,80)
100 LINE(128,140)-(20,80)
110 LINE(128,25)-(30,80)
120 LINE(128,25)-(163,45)
130 LINE(190,60)-(225,80)
140 LINE(128,135)-(225,80)
150 LINE(30,80)-(65,100)
160 LINE(93,115)-(128,135)
170 LINE(153,50)-STEP(10,-5)
180 LINE(65,100)-STEP(24,-13)
190 LINE(109,75)-STEP(35,-19)
200 LINE(117,101)-(190,60)
210 LINE(93,115)-STEP(10,-6)
220 CIRCLE(128,80),38,,3.14/2,3*3.14/2
230 CIRCLE(128,80),28,,3.14/2,3*
                              3.14/2,1.5
240 CIRCLE(130,60),28,,0,3.16/2,2/3
250 CIRCLE(130,60),18,,0,3.16/2,.5
260 CIRCLE(130,100),28,,3.1*3/2,0,2/3
270 CIRCLE(130,100),18,,3*3/2,0,.5
280 LINE(148,60)-STEP(9,0)
290 LINE(148,100)-STEP(9,0)
300 PAINT(128,11)
310 PAINT(128,26)
320 PAINT(128,134)
330 PAINT(128,11)
340 PAINT(128,58)
350 COLOR 12,1
360 PSET(0,170),15
370 PRINT #1,"          СПАРТАК - ЧЕМПИОН"
380 PSET(0,171),15
390 PRINT #1,"          СПАРТАК - ЧЕМПИОН"
400 PLAY "C8 R16 C8 R16 C16 R16 C16 R16
        C16 R8 C16 R16 C16 R16 C16 R16
                  C16 R8 C16 R16 C8 R1"
410 GOTO 400
```

Устойчивый интерес к определенному эффекту, возникший у учащихся, позволяет при изучении новых тем периодически возвращаться к программам, основанным на данном эффекте. Ниже приводится совокупность заданий, позволяющих получить на экране быстро сменяющую друг друга индикацию последовательных чисел («секундомер»).

УЧЕБНАЯ ЦЕЛЬ	ФОРМУЛИРОВКА ЗАДАНИЯ	РЕШЕНИЕ
Освоение оператора цикла	Печатайте значения параметра цикла в одном и том же месте чистого экрана (исходная форма задания)	10 CLS 20 FOR I=1 TO 1000 30 LOCATE 15,10 40 PRINT I 50 NEXT I
Применение функции INT	Замедлите секундомер	10 CLS 20 FOR I=1 TO 1000 30 LOCATE 15,10 40 PRINT INT(I/10) 50 NEXT I
Подготовка к написанию программ суммирования	Создайте зацикленный секундомер,используя оператор присваивания	10 CLS:I=0 20 I=I+1 30 LOCATE 15,10 40 PRINT INT(I/10) 50 GOTO 50
Освоение вложенных циклов	Замедлите секундомер с помощью пустого цикла	10 CLS 20 FOR I=1 TO 1000 30 LOCATE 15,10 40 PRINT I 50 FOR T=1 TO 100 55 NEXT T 60 NEXT I
Освоение вложенных циклов с переменными границами для параметров	Заставьте секундомер постепенно замедляться	10 CLS 20 FOR I=1 TO 1000 30 LOCATE 15,10 40 PRINT I 50 FOR T=1 TO I 55 NEXT T 60 NEXT I
Освоение условного оператора	Добавьте к секундомеру индикацию минут	10 CLS 20 J=0:I=0 30 IF I=60 THEN J=J+1 35 I=0 40 LOCATE 10,10 50 PRINT J;"мин ";I; "сек" 60 FOR T=1 TO 300 65 NEXT T 70 I=I+1:GOTO 30

В данном случае все программы предельно просты, и изменения в них при переходе к новым темам минимальны. Возможны иные варианты возвращения к уже знакомым учащимся заданиям, содержащим определенные приемы программирования и вызывающим интересные эффекты. Приведем два примера. Комбинирование эффектов и приемов

ИСХОДНЫЕ ЗАДАНИЯ	РЕШЕНИЕ
1. Нарисуйте забавную рожицу.	10 COLOR 10,1,1:SCREEN 2 20 CIRCLE(128,96),80 30 CIRCLE(88,90),20 40 CIRCLE(88,90),15 50 CIRCLE(168,90),20 60 CIRCLE(168,90),15 70 CIRCLE(88,90),10 80 CIRCLE(168,90),10 90 CIRCLE(128,120),15 100 CIRCLE(128,136),20,,3.14,0 110 CIRCLE(128,136),20,,3.14,0,.5 120 PAINT(128,150) 130 PAINT(88,76) 140 PAINT(168,76) 150 GOTO 150
2. Заставьте окружность надуваться, как воздушный шарик	10 COLOR 10,1,1:SCREEN 2 20 CIRCLE(95,120),10 30 FOR R=10 TO 70 40 CIRCLE(95,120),R+1 50 CIRCLE(95,120),R,1 60 NEXT R 70 GOTO 70

3. Напишите программу, которая заставит летать взад вперед по диагонали какой-либо символ	10 CLS 20 A=0:B=20:S=1 30 FOR I=A TO B STEP S 40 LOCATE I,I 50 PRINT "X" 60 FOR T=0 TO 100:NEXT T 70 CLS 80 NEXT I 90 SWAP A,B:S=-S 100 GOTO 30

КОМБИНИРОВАННОЕ ЗАДАНИЕ	РЕШЕНИЕ
Изобразите рожицу с попеременно расширяющимися и сужающимися глазами	10 COLOR 10,1,1:SCREEN 2 20 CIRCLE(128,96),80 30 CIRCLE(88,90),20 40 CIRCLE(88,90),15 50 CIRCLE(168,90),20 60 CIRCLE(168,90),15 70 CIRCLE(88,90),10 80 CIRCLE(168,90),10 90 CIRCLE(128,120),15 100 CIRCLE(128,136),20,,3.14,0 110 CIRCLE(128,136),20,,3.14,0,.5 120 PAINT(128,150) 130 PAINT(88,76) 140 PAINT(168,76) 150 R1=20:R2=30:S=1 160 FOR R=R1 TO R2 STEP S 170 CIRCLE(88,90),R 180 CIRCLE(168,90),R 190 CIRCLE(88,90),R-S,1 200 CIRCLE(168,90),R-S,1 210 NEXT R 220 SWAP R1,R2:S=-S 230 GOTO 160

Изменение эффекта работы программы с помощью подключения новых средств языка программирования

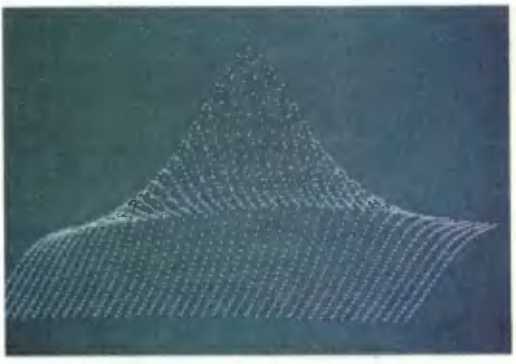
ИСХОДНОЕ ЗАДАНИЕ	РЕШЕНИЕ
Построить изображение поверхности, заданной уравнением z = F(x,y) В результате будет построена поверхность, изображение которой содержит невидимые точки (рис.2)	10 SCREEN 2 20 FOR Y=0 TO 45 30 FOR X=0 TO 200 STEP 5 40 GOSUB 100 50 XT=X+Y 60 YT=191-Y-ZT 70 PSET (XT,YT) 80 NEXT X,Y 90 GOTO 90 100 XF=(X-100)/10 110 YF=(Y-27)/2 120 ZT=350/SQR(XF^2+YF^2+7) 130 RETURN

ПОВТОРНОЕ ЗАДАНИЕ, ПРЕДПОЛАГАЮЩЕЕ ИСПОЛЬЗОВАНИЕ МАССИВА	РЕШЕНИЕ
Построить изображение той же поверхности, устранив невидимые точки (рис.3)	10 DIM M(255) 20 SCREEN 2 30 FOR Y=0 TO 45 40 FOR X=0 TO 200 STEP 5 50 GOSUB 120 60 XT=X+Y 70 YT=191-Y-ZT 80 IF M(XT)<ZT THEN M(XT)=ZT 90 IF M(XT)<=ZT THEN PSET(XT,YT) 100 NEXT X,Y 110 GOTO 110 120 XF=(X-100)/10 130 YF=(Y-27)/2 140 ZT=350/SQR(XF^2+YF^2+7) 150 RETURN

Учащимся, осваивающим информатику быстрее, чем предусмотрено учебным планом, следует предлагать усложненные задания по изучаемой теме. Однако если основные задания направлены на освоение операторов языка программирования, то дополнительные должны стимулировать выдумку и творческий подход к предмету. Приведем несколько таких примеров, связанных с использованием работы в прямом диалоге (см. таблицу).

Приведем пример использования прямого диалога при изучении темы «Построение графиков».

Задание. При построении графиков функций довольно большое время уходит на вычисление каждой ординаты. Чтобы ускорить построение, можно заранее вычислить все ординаты, записать их в операторах DATA и считывать опе-



2

3

Изучаемая тема	Задание	Решение
Работа в режиме калькулятора	Организовать вычисление последовательных степеней тройки с помощью самокорректирующейся командной строки	Использовать командную строку LOCATE 20,0:PRINT 3×1, которую следует набрать на чистом экране в верхней строке экрана. Результат печатается в этой же экранной строке после знака умножения. Для ускорения работы устанавливаем новое значение функциональной клавиши F1: KEY 1, CHR$ (13)+ +CHR$ (30)+CHR$ (30). Код 30 соответствует переводу курсора строкой выше. Нажатие на клавишу F1 одновременно запускает командную строку, которая себя корректирует и возвращает в нее курсор
Работа в режиме калькулятора	Разработайте способ быстрого вычисления произведения цифр числа. Проверьте его на примере числа 7^11	Вычисляем число командой PRINT 7 ^ 11. Вставляем перед результатом знак вопроса (вместо PRINT). Вставляем знак умножения, используя клавишу F2 со значением, установленным командой KEY 2, CHR$ (28)+CHR (18)+«×»
Изучение графических операторов	Разработайте прием, позволяющий работать с графикой в прямом диалоге (графический режим отключается сразу после исполнения командной строки, что не дает возможности задержать рисунок на экране)	Установим новое значение клавиши F5 командой KEY 5, «:RUN»+CHR$ (13). Наберем программу 10 IF INKEY$=""GOTO 10 20 END После набора командной строки, например SCREEN 2:CIRCLE (120, 90), 95, достаточно нажать на F5

ратором READ. Автоматизируйте процесс вычислений и создания программных строк, содержащих операторы DATA.

Решение. Наберем программу, содержащую в строках 240—300 блок для построения графика. Со строки 1000 наберем программу-генератор. Она вычисляет ординаты и печатает на экране содержащие их программные строки. Их номера определяются переменной L. При первом запуске задаем ей значение, равное 0. В дальнейшем нужное значение подпечатывается на экране в том месте, где оно будет запрошено оператором ввода. После выдачи очередной порции строк на экран нужно несколько раз нажать на клавишу ввода, чтобы ввести новые программные строки, снова запустить программу и ввести новое значение переменной L. Когда новая 240-я строка забьет старую, будет построен график.

```
240 GOTO 1000
250 SCREEN 2
260 FOR X=0 TO 249
270 READ Y
280 PSET (X,Y)
290 NEXT X
300 GOTO 300
1000 '  *** ПРОГРАММА - ГЕНЕРАТОР ***
1010 DEF FNF(X)=SIN(2*X)*2+SIN(5*X)/2
1020 LOCATE0,19
1030 INPUT L
1040 CLS:LOCATE,1
1050 FOR I=L TO L+40 STEP 10
1060 PRINT I;"DATA";
1070 FOR X=I TO I+9
1080 P=INT(96-20*FNF(X/20))
1090 S$=STR$(P)
1100 PRINT MID$(S$,2,LEN(S$)-1);
1110 IF INT((X-9)/10)<>(X-9)/10 THEN
                                PRINT ",";
1120 NEXT X
1130 PRINT :PRINT
1140 NEXT I
1150 PRINT "run"
1160 LOCATE,19:PRINT " ";I
1170 LOCATE 0,0
```

Сильных учащихся программы, связанные с компьютерной графикой, привлекают так же, как и всех остальных. Им можно предложить программы, основывающиеся на использовании спрайтов, файлов, простых подпрограмм в машинных кодах.

Задание. Создайте спрайты, соответствующие всем буквам латинского алфавита, и заставьте их двигаться по экрану.

Решение

```
10  DIM K(25)
20 X=RND(-TIME)
```

```
30 SCREEN 1,1
40 COLOR 15,1,1
50 KEY OFF
60 LOCATE,10
70 PRINT "  ВЫБИРАЕТСЯ ПОРЯДОК БУКВ  "
80 'БУКВЕННЫЕ СПРАЙТЫ''''''''''''''''''
90 N1=ASC("A")                          :'
100 N2=ASC("Z")                         :'
110 FOR N=N1 TO N2                      :'
120 Z$=""                               :'
130 FOR J=0 TO 7                        :'
140 Z$=Z$+CHR$(VPEEK(N*8+J))            :'
150 NEXT J                              :'
160 SPRITE$(N-N1)=Z$                    :'
170 'ПОРЯДОК ПОЯВЛЕНИЯ БУКВ''''''       :'
180 K(N-N1)=INT(26*RND(1))      :'      :'
190 IF N-N1=0 GOTO 240          :'      :'
200 FOR J=0 TO N-N1-1           :'      :'
210 IF K(J)=K(N-N1) GOTO 180    :'      :'
220 NEXT J:''''''''''''''''''''''''     :'
230 '                                   :'
240 NEXT N:''''''''''''''''''''''''''''''
250 '
260 'ДЕМОНСТРАЦИЯ ДВИЖЕНИЯ СПРАЙТОВ'''
270 SCREEN 2                            :'
280 OPEN "GRP:" AS #1                   :'
290 LINE (0,0)-(255,191),15,B           :'
300 DATA "Огурцы в кадушке,"            :'
310 DATA "Мысли в голове,"              :'
320 DATA "Перышки в подушке,"           :'
330 DATA "Соловьи в листве."            :'
340 DATA ""                             :'
350 DATA "     Уолтер де Ла Мэр"        :'
360 FOR I=1 TO 6                        :'
370 PRESET(60,I*10+60)                  :'
380 READ S$                             :'
390 PRINT #1,S$                         :'
400 NEXT I                              :'
410 DIM X(25),Y(25),DX(25),C(25)        :'
420 FOR I=0 TO 25                       :'
430 X(I)=I*9+10                         :'
440 Y(I)=I*7                            :'
450 DX(I)=INT(RND(1)*8)*2-7             :'
460 C(I)=K(I)MOD14+2                    :'
470 NEXT I                              :'
480 FOR I=0 TO 25                       :'
490 PUT SPRITE I,(X(I),Y(I)),C(I),K(I)
500 X(I)=(X(I)+DX(I))MOD256             :'
510 NEXT I                              :'
520 GOTO 480:''''''''''''''''''''''''''
```

В программе используется тот факт, что форма любого символа в режиме SCREEN 1 определяется содержимым восьми байтов видеопамяти, начиная с байта с номером 8*N, где N — код данного символа в стандарте ASCII.

Задание. В программе требуется использовать спрайты, в совокупности изображающие комету. Придумайте способ, позволяющий достаточно быстро получить спрайты, изображающие хвост кометы.

Р е ш е н и е.

```
10 COLOR 10,1,1:SCREEN 2
20 FOR K=2 TO .5 STEP -.1
30 CIRCLE(63-22/3*K+44/3,50),20+4*K/2.5
                          ,,3.14/2,3.14,K
40 NEXT K
45 LINE(49,27)-STEP(26,23),,B
50 FOR X=49 TO 65 STEP 8
60 FOR Y=27 TO 42
70 B$="&b"
80 FOR I=0 TO 7
90 IF POINT(X+I,Y)=10 THEN B$=B$+"1"
                      ELSE B$=B$+"0"
100 NEXT I
110 LPRINT VAL(B$);
120 NEXT Y,X
```

Изображение хвоста кометы рисуется с помощью семейства дуг эллипсов с различными коэффициентами сжатия. Затем соответствующая область экрана разбивается на вертикальные полосы шириной 8 точек. Изображение, занимающее 8 точек по горизонтали, преобразуется в число, которое соответствует горизонтальному элементу изображения создаваемого спрайта. Числа печатаются на бумаге.

Задание. В графическом режиме нарисован одноцветный рисунок. Придумайте способ, позволяющий быстро сменить цвет рисунка и фон.

Р е ш е н и е. Для смены цветов без уничтожения рисунка достаточно изменить значение всех байтов таблицы цветов (начинается с адреса & H2000 видеопамяти и содержит & H1800 байтов). Быстрота обеспечивается использованием подпрограммы BIOS с адресом & H56, заполняющей интервал видеопамяти определенным значением.

```
10 DATA &h21,&h00,&h20,&h01,&h00,&h18,
     &h3A,&hFF,&h8F,&hCD,&h56,&h00,&hC9
20 FOR I=&H8FF0 TO &H8FFC
30 READ B
40 POKE I,B
50 NEXT I
60 DEF USR=&H8FF0
70 ''''''''''''''''''''''
80 SCREEN 2
90 FOR X=15 TO 240 STEP 25
100 FOR Y=13 TO 178 STEP 15
110 FOR R=6 TO 16 STEP 5
120 CIRCLE(X,Y),R,,,,.5
130 NEXT R
140 NEXT Y
150 NEXT X
160 ''''''''''''''''''''''
170 FOR I=2 TO 15
180 POKE &H8FFF,I
190 Z=USR(Z)
200 FOR T=1 TO 500
210 NEXT T
220 NEXT I
230 FOR I=2 TO 15
240 POKE &H8FFF,I*16
250 Z=USR(Z)
260 FOR T=1 TO 500
```

```
270 NEXT T
280 NEXT I
290 GOTO 170
```

Для подготовки данных для стандартной подпрограммы используется подпрограмма в машинных кодах. Коды расположены в строке 10 и загружаются в физическую память компьютера в строках 20—50. Значение, которым заполняется таблица цветов, предварительно заносится в байт с адресом & H8FFF. Приведем текст вспомогательной подпрограммы на ассемблере.

```
LD HL ,&H2000
LD BC ,&H1800
LD A  ,(&H8FFF)
CALL 0056
RET
```


Задание. Напишите программу, позволяющую сохранять на диске заранее построенное графическое изображение и загружать его в видеопамять с диска.

Р е ш е н и е.

```
10 OPEN "grp:" AS #1:COLOR 10,1,1
20 SCREEN 2,0
30 PSET(128,6)
40 FOR I=1 TO 101
50 X=128+90*SIN(I*3.14*87/101)
60 Y=96-90*COS(I*3.14*87/101)
70 LINE-(X,Y)
80 NEXT I
90 DATA &H21,&H00,&H00,&H11,&H00,&H90,
     &H01,&H00,&H18,&HCD,&H59,&H00,&HC9
100 FOR I=0 TO 12
110 READ R
120 POKE &H8FF0+I,R
130 NEXT I
140 DEF USR=&H8FF0
150 Z=USR(Z)
160 POKE &H8FF2,&H20
170 POKE &H8FF5,&HA8
180 Z=USR(Z)
190 BSAVE "картинка",&H9000,&HC000
200 '''''''''''''''''''''''''''''''''''''''
210 LINE(0,0)-(255,191),1,BF
220 DATA &h21,&H00,&H90,&H11,&H00,&H00,
     &H01,&H00,&H18,&HCD,&H5C,&H00,&HC9
230 FOR I=0 TO 12
240 READ R
250 POKE &H8F00+I,R
260 NEXT I
270 DEF USR=&H8F00
280 BLOAD "картинка"
290 Z=USR(Z)
300 POKE &H8F02,&HA8
310 POKE &H8F05,&H20
320 Z=USR(Z)
330 GOTO 330
```

Загрузка картинки на диск и считывание с диска производятся с помощью команд BSAVE и BLOAD. Однако поскольку эти команды работают с памятью, а картинка хранится в видеопамяти, то необходимо сначала перенести соответствующие области видеопамяти в память, а затем производить запись на диск. После считывания информации с диска в память необходимо перенести его в видеопамять. Для этой цели используются подпрограммы BIOS с адресами & H59 и & H5C.

Наконец перейдем к заданиям, которые преподаватель может предлагать учащимся на завершающем этапе обучения. Из большого количества разнообразнейших заданий можно выделить группу стандартных заданий: тесты, простейшие модели, инструментальные программы и т. д.

Задание. Напишите простую тестирующую программу (шуточную).

Р е ш е н и е.

```
10 DIM W$(3),OT$(7):KD$="&B"
20 W$(1)="любовь ?"
30 W$(2)="дружбу ?"
40 W$(3)="счастье ?"
50 OT$(0)="Вы ДЕМОНИЧЕСКАЯ ЛИЧНОСТЬ"
60 OT$(1)="Вы расчетливый
                          ИНДИВИДУАЛИСТ"
70 OT$(2)="У вас низкий уровень
                          ИНТЕЛЛЕКТА"
80 OT$(3)="Вы скрытый ПЕССИМИСТ"
90 OT$(4)="Вы склонны к ИСТЕРИИ"
100 OT$(5)="Вы ЧУВСТВЕННАЯ НАТУРА"
110 OT$(6)="Вы страдающий АЛЬТРУИСТ"
120 OT$(7)="Вы слишком ДОВЕРЧИВЫ"
130 CLS:KEY OFF
140 LOCATE ,2
150 PRINT "  ПРИ ОТВЕТЕ НАЖАТИЕ НА
                          КЛАВИШУ 'Д'
160 PRINT
170 PRINT "  ОЗНАЧАЕТ 'ДА';НАЖАТИЕ НА
                          ЛЮБУЮ
180 PRINT
190 PRINT "  ДРУГУЮ КЛАВИШУ
                          ОЗНАЧАЕТ 'НЕТ'
200 LOCATE ,10
210 FOR I=1 TO 3
220 PRINT "Верите ли вы в ";
230 PRINT W$(I);"      - ";
240 F$=INPUT$(1)
250 IF F$="d" OR F$="D" OR F$="д" OR
     F$="Д" THEN KD$=KD$+"1":PRINT
     "ДА" ELSE KD$=KD$+"0":PRINT "НЕТ"
260 PRINT
270 NEXT I
280 PRINT
290 K=VAL(KD$)
300 L=LEN(OT$(K)):N=INT((39-L)/2)-2
310 LOCATE N:PRINT STRING$(L+4,"*")
320 LOCATE N
```

```
330 PRINT "* ";OT$(K);" *"
340 LOCATE N:PRINT STRING$(L+4,"*")
350 LOCATE ,22
360 END
```

Поскольку тесты очень популярны среди учащихся, подобные программы, предполагающие самостоятельную разработку вопросов и результатов тестирования, могут быть предложены широкому кругу учащихся. Задания на написание простых моделирующих программ предъявляют большие требования к математической подготовке учащихся.

Задание. Напишите программу, моделирующую одностороннее движение по шоссе. Промежуток времени между автомашинами является случайным числом. Скорость всех автомашин одинакова.

Р е ш е н и е.

```
10 COLOR 10,1,1:SCREEN 2,0
20 SPRITE$(0)=STRING$(8,CHR$(248))
30 LINE(120,0)-(134,191),7,BF
40 LM=.02
50 DIM Y(30)
60 Z=RND(-TIME)
70 N=0:L=0:Y(0)=-7
80 SOUND 7,&B10110111:SOUND 8,15
90 FOR I=N TO N+L
100 IM=IMOD31
110 Y(IM)=Y(IM)+2
120 NEXT I
130 IF Y(N)>191 THEN PUT SPRITE
                     N,(-8,-8),1,0:Y(N)=0:
                       N=(N+1)MOD31:L=L-1
140 K=(N+L)MOD31
150 IF Y(K)>-2 AND Y(K)<2 THEN L=L+1
160 K=(K+1)MOD31
170 Y(K)=-INT(-LOG(1-RND(1))/LM)-9
180 FOR I=N TO N+L
190 IM=IMOD31
200 IF Y(IM)>-8 THEN PUT SPRITE
                       IM,(125,Y(IM)),6,0
210 NEXT I
220 IF INKEY$="" THEN 90 ELSE BEEP
```

Перед тем как приступить к работе над программой, учащийся должен ознакомиться со способами получения случайных чисел с различными функциями распределения из случайных чисел, равномерно распределенных на отрезке. В приведенной программе используется распределение, задаваемое функцией

$$F(R)=1-e^{-\lambda R}.$$

Для получения случайных чисел с нужным распределением используется формула

$$R=-\frac{\ln(1-X)}{\lambda},$$

где X — случайное число с равномерным распределением на отрезке (0,1).

Некоторые задания могут предполагать участие нескольких учащихся в написании итоговой программы. При таком подходе одно большое задание разбивается на ряд мелких, каждое из которых выполняется одним исполнителем. Затем производится совместная стыковка блоков.

Общее задание. Напишите программу, которая выдает на экран случайную гладкую линию. Линия предназначена для стимуляции творческого воображения пользователя-рисовальщика. В его распоряжении должны находиться графический редактор, позволяющий дорисовывать детали изображения на экране, и средство печати изображения на бумаге.

Разбиение вытекает из содержания задания. Программа должна иметь три блока:

1. Генератор случайной линии.
2. Графический редактор.
3. Блок печати изображения на бумаге.

Не останавливаясь на блоке 2, приведем небольшие эскизы блоков 1 и 3.

Генератор случайной линии может ее строить с помощью сопряженных дуг окружностей случайного радиуса.

```
10 X=RND(-TIME)
20 X0=128
30 Y0=96
40 R0=30*RND(1)+3
50 A0=6.28*RND(1)
60 B0=6.28*RND(1):B=B0:S=1
70 SCREEN 2
80 CIRCLE(X0,Y0),R0,,A0,B0
90 B0=B
100 IF RND(1)<.5 THEN E=-1 ELSE E=1
110 R1=30*RND(1)+3
120 B1=6.28*RND(1)
130 B=B1
140 X1=X0+(R0+E*R1)*COS(B0)
150 Y1=Y0-(R0+E*R1)*SIN(B0)
160 A1=B0+3.14*(E+1)/2
170 IF A1>6.28 THEN A1=A1-6.28
180 IF S*E>0 THEN SWAP A1,B1:S=-1
                                ELSE S=1
190 X0=X1:Y0=Y1:R0=R1:A0=A1:B0=B1
200 FOR I=1 TO 100:NEXT
210 IF INKEY$="" THEN 80
220 GOTO 10
```

Для блока печати изображений приведем два варианта. Первый из них воспроизводит экранную точку на бумаге одним ударом штырька печатающей головки, второй — четырьмя (2 на 2). И в том, и в другом случае изображение повернуто на 90°.

Вариант 1

```
10 SCREEN 2
20 FOR X=-80 TO 256+80 STEP 8
30 CIRCLE (X,96),90
40 NEXT X
50 POKE &HF41B,1
60 LPRINT CHR$(27);"T16";
70 FOR A=0 TO 248 STEP 8
80 LPRINT CHR$(27);"S0192";
90 FOR D=0 TO 23
100 FOR I=0 TO 7
110 KD=VPEEK(A+D*256+I)
120 LPRINT CHR$(KD);
130 NEXT I,D
140 LPRINT
150 NEXT A
```

Вариант 2

```
10 DATA 0,3,12,15,48,51,60,63
20 DATA 192,195,204,207,240,243,252,255
30 DIM DB(15),TS(255)
40 FOR I=0 TO 15
50 READ DB(I)
60 NEXT I
70 SCREEN 2
80 FOR X=-80 TO 256+80 STEP 8
90 CIRCLE(X,96),90
100 NEXT X
110 POKE &HF41B,1
120 LPRINT CHR$(27);"T16";
130 FOR A=0 TO 248 STEP 8
140 LPRINT CHR$(27);"S0384";
150 FOR D=0 TO 23
160 FOR I=0 TO 7
170 KD=VPEEK(A+D*256+I)
180 Q=INT(KD/16)
190 R=KD-16*Q
200 LPRINT CHR$(DB(Q));CHR$(DB(Q));
210 TS(D*8+I)=DB(R)
220 NEXT I,D
230 LPRINT
240 LPRINT CHR$(27);"S0384";
250 FOR J=0 TO 191
260 LPRINT CHR$(TS(J));CHR$(TS(J));
270 NEXT J
280 LPRINT
290 NEXT A
300 END
```

Литература

1. Практические занятия на ПЭВМ «Ямаха» / Под ред. Антипова И. Н. М.: 1987. Изд-во НИИ школ МП РСФСР.

В. РУДЕНКО
канд. пед. наук, НИИ педагогики УССР

Имитационное моделирование учебных алгоритмов

Одной из наиболее сложных и трудно усваиваемых тем школьной информатики является алгоритмизация. Для глубокого усвоения принципов построения и функционирования алгоритма недостаточно правильно его изобразить в виде блок-схемы или описать на алгоритмическом языке. На наш взгляд, требуется и пояснение методики исполнения алгоритма при конкретных исходных данных. Но объяснение функционирования алгоритма достаточно трудоемко и требует значительного времени. Поэтому методика исполнения алгоритма часто излагается упрощенно. К тому же используемые для этого приемы и средства не отличаются высокой наглядностью и требуют развитого абстрактного мышления как педагога, так и учащихся.

Наглядность разработки и функционирования учебных алгоритмов достигается при использовании метода имитационного моделирования. В научных исследованиях этот метод используется для имитации на ЭВМ процесса функционирования алгоритмов с целью определения правильности проекта и его качества. В учебном процессе использование имитационного моделирования определяется целями и задачами обучения. Как мы уже отмечали, главная цель имитации создания и исполнения алгоритмов на уроках информатики — углубленное понимание учащимися этих процессов.

При использовании метода имитационного моделирования алгоритма можно детализировать в различной степени и

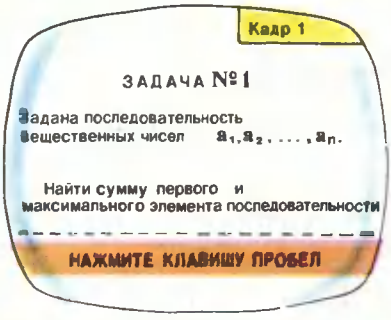
использовать любые извесные способы его описания. Однако наиболее высокая наглядность демонстрации процесса создания и функционирования алгоритма достигается при использовании блок-схем.

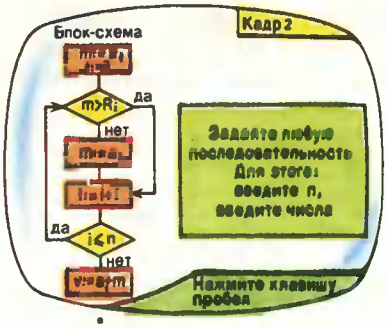
Сущность предлагаемой методики состоит в следующем. Постановка задачи на разработку алгоритма формулируется учителем и может одновременно отображаться на экранах дисплеев учащихся. Далее учитель объясняет метод решения задачи. Это разъяснение сопровождается автоматическим построением блок-схемы алгоритма на ученических дисплеях. Скорость построения блок-схемы определяет обучаемый. После этого учащимся предлагается самостоятельно задать исходные данные. По окончании их ввода начинается моделирование работы алгоритма. При этом весь процесс функционирования алгоритма для введенных исходных данных в замедленном темпе отображается на экране дисплея. Темп воспроизведения может выбирать обучаемый, которому предоставляется возможность в любой момент приостановить или продолжить исполнение алгорима, изменить исходные данные и осуществить повторное моделирование.

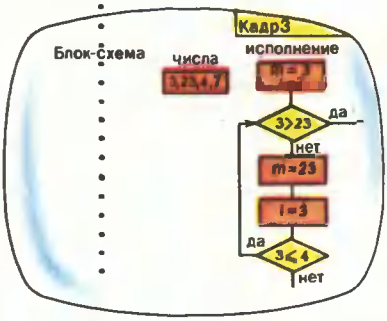
Описанная методика реализуется двумя способами. Ниже приводится пример, поясняющий принцип использования первого способа имитационного моделирования учебных алгоритмов.

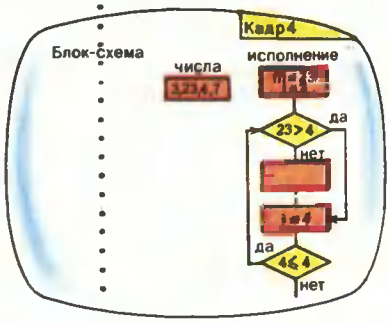
На экране дисплея последовательно появляется информация, содержащаяся в кадрах $1 \div 5$.

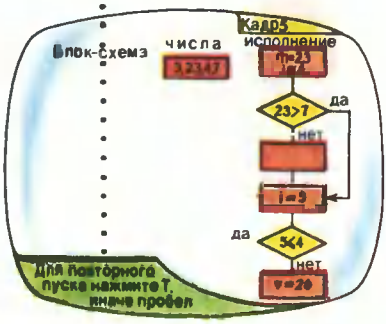
Динамика появляения информации в кадре 2 следующая. Вначале отображается блок-схема алгоритма. В программе предусмотрено три скорости ее воспроизведения. После нажатия клавиши











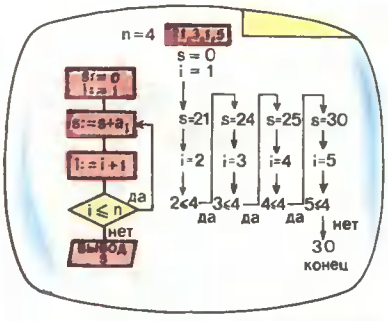
ПРОБЕЛ на экране высвечивается требование ввода исходных данных. Ввод осуществляется последовательно, число за числом. Вначале вводится n, затем a_1, a_2 и т. д. до a_n. Пусть, например, учащийся ввел $n=4$ и $a_1=3$, $a_2=23$, $a_3=4$, $a_4=7$. После повторного нажатия клавиши ПРОБЕЛ появится кадр 3.

Динамика воспроизведения информации в кадре 3 следующая. В блок-схеме* последовательно в замедленном темпе в порядке выполнения выделяются другим цветом (или фоном) соответствующие блоки и линии соединения. Синхронно изображаются блоки с соответствующей информацией и соединительные линии алгоритма исполнения. Спустя некоторое время (примерно 10 с) после завершения изображения кадра 3 следует автоматическая смена кадров и в аналогичной последовательности воспроизводится информация, изображенная на кадрах 4 и 5.

После демонстрации кадра 5 при нажатии на клавишу Т осуществляется переход на кадр 1, а при нажатии на клавишу ПРОБЕЛ — вход в систему.

При втором способе моделирование исполнение алгоритма осуществляется только на одной странице экрана монитора. На рис. 6 изображена блок-схема алгоритма нахождения суммы членов линейной последовательности чисел a_1, a_2, ..., a_n, а также исполнения алгоритма для чисел 21, 3, 1, 5. Второй способ, вероятно, обладает более высокой наглядностью, чем первый, так как весь процесс исполнения алгоритма обозревается как единое целое на одном экране.

6



* Блок-схема в кадрах 3, 4, 5 та же, что и в кадре 2.— *Прим. ред.*

Однако этим способом можно осуществлять имитацию работы алгоритмов, на исходные данные которых накладываются определенные ограничения (моделирование исполнения алгоритма, изображенного на рис. 6, можно осуществить лишь для линейных последовательностей, содержащих не более четырех чисел). В первом варианте таких ограничений практически не существует.

Мы рассмотрели два способа имитационного моделирования учебных алгоритмов. Конечно, возможны и другие. Выбор того или иного способа должен определяться содержанием решаемой задачи, а основным критерием выбора должна быть наглядность демонстрации исполнения алгоритма.

Отметим, что возможности учебных ПЭВМ (например, «Корвет», «Ямаха») накладывают определенные ограничения на применение метода имитационного моделирования. Целесообразно, чтобы количество символов блок-схемы алгоритма, одновременно воспроизводимых на дисплее, не превышало 12. Оптимальными следует считать следующие размеры символа: по горизонтали — 40 мм, по вертикали — 20 мм (что соответствует для «Ямахи» 40—20 пикселей). Количество блоков по вертикали не должно превышать 6, а расстояние между блоками п вертикали — 10 мм.

В ходе экспериментальной работы, осуществляемой в НИИ педагогики УССР, мы пришли к выводу, что целесообразно разрабатывать и использовать моделирующие программы по следующей тематике.

1. Алгоритмы поиска максимального и минимального элементов в линейных неупорядоченных числовых последовательностях.
2. Алгоритмы поиска элемента по ключу запроса в линейных упорядоченных числовых последовательностях.
3. Алгоритмы обработки линейных числовых последовательностей.
4. Алгоритмы преобразования элементов числовой последовательности.
5. Алгоритмы перемещения элементов последовательности.
6. Алгоритмы упорядочения числовых последовательностей.
7. Алгоритмы формирования новых

числовых последовательностей.

8. Алгоритмы обработки двумерных числовых массивов.

9. Алгоритмы поиска элементов в двумерном массиве.

1. Разветвляющиеся вычислительные алгоритмы.

11. Циклические вычислительные алгоритмы.

12. Циклические вычислительные алгоритмы с массивами.

Опыт разработки моделирующих программ на уроках информатики подтверждает высокую эффективность использования имитационного моделирования.

Литература

1. Энциклопедия кибернетики. Т. 2. Киев, 1975.

2. Имитационное моделирование на ЭВМ. М.: Радио и связь, 1988.

В. КУЗНЕЦОВ
г. Курган

Как войти в информатику?

Вопросу принципиальной важности: с чего и как начать обучение информатике — до сих пор не уделяется должного внимания. Нет ничего более деликатного и мимолетного, чем начало, которое во многом определяет дальнейшее развитие знаний. Даже наличие компьютера с комплектом игр, вызывая первоначальное любопытство и азарт, всего лишь отодвигает решение этой серьезной проблемы. Не решают ее, помоему, и существующие учебные пособия, которые, спеша превратить учащегося в программиста, объединяют четыре слабо увязанные между собой и трудно интегрируемые начинающими направления: 1) интуитивное понятие алгоритма как последовательности предписаний некоторому абстрактному исполнителю, характеризуемому своей системой команд, с иллюстрациями в виде примеров «из жизни»; 2) алгоритмическая нотация как модель реальных языков программирования; 3) конкретизация принципов работы ЭВМ путем фрагментарного описания архитектуры реальных микро-ЭВМ; 4) примеры применения ЭВМ в различных областях человеческой деятельности.

Неизвестно, сколько и каких примеров нужно привести, чтобы быть уверенным в корректном формировании понятия алгоритма, а отсутствие целостных представлений у учащихся об ЭВМ быстро проявляется в вузе, где новая операционная обстановка, лекционно-лабораторная система занятий и установившаяся методика обучения порой не согласуются с приобретенными в школе знаниями. «Выживают» лишь те немногие, кто еще в школе самостоятельно в различных кружках перепробовал разные ЭВМ, разные операционные системы и языки программирования.

В свете обозначенной проблематики представляется, что первоначальный этап обучения должен удовлетворять следующим условиям.

1. Формировать целостное, сбалансированное и достаточно определенное понятие алгоритма (единство физического и логического описания исполнителя).

2. Это понятие должно быть достаточно примитивным: легко охватываемым и вынуждающим развивать самое себя.

3. Процесс обучения строится как последовательность конкретизирующих уточнений, направленных к современному состоянию информатики.

Таким образом может быть достигнуто «концептуальное единство» курса информатики.

Итак, вопрос сводится к уточнению понятия алгоритма, а учитывая его фундаментальность (в смысле несводимости к другим математическим структурам) — к принятию некоторой точной

и по возможности наипростейшей модели. Разумеется, математика (все-таки математика!) давно уже выработала такого рода модели — машины Тьюринга, Поста, исчисления Черча, Маркова и др. [1]. Однако эти классические модели родились в докомпьютерную эпоху из внутренних математических потребностей и, естественно, «концептуальное расстояние» от них до современных ЭВМ оказывается слишком большим и требующим дополнительных и не совсем оправданных интеллектуальных шагов. Наиболее адекватная фон неймановской архитектуре модель, строго определяющая класс допустимых данных, способ хранения и доступ к ним, систему операций над данными и удовлетворяющая сформулированным условиям, родилась намного позже (как это обычно бывает в развитии науки) — в 1963 г. Это так называемая машина с неограниченными регистрами — МНР [2]. Именно эту модель целесообразно положить в основу курса информатики, а ее изящество и простота заслуживают более развернутого рассмотрения.

МНР оперирует с натуральными числами, которые хранятся в ячейках (регистрах) однородной дискретной памяти. Одна ячейка может хранить любое натуральное число (т. е. емкость ячейки бесконечна — это ведь теоретическая модель), количество ячеек не ограничено, все ячейки перенумерованы числами натурального ряда. МНР-программа, использующая команды из предустановленной системы, представляет собой некоторый список команд, начинает свою работу с первой записанной команды и заканчивает работу по исчерпании команд. Система команд МНР (см. таблицу.) содержит 6 команд, которые, кроме четвертой, не нарушают естественного порядка выполнения команд в программе, а четвертая осуществляет условный переход к команде с заданным номером при совпадении содержимого двух заданных ячеек. Команды ввода-вывода информации, реализующие связь памяти с внешним миром, добавлены для большей реалистичности модели.

Более детальное знакомство с МНР можно начать с составления программы сложения двух натуральных чисел:

```
1I(1)
2I(2)
3Z(3)
4J(2, 3, 8)
5S(1)
6S(3)
7J(1, 1, 4)
80(1)
```

в которой как в капле воды отражается почти весь арсенал приемов программирования. Тут и поиск метода решения путем добавления к одному из слагаемых (в 1-й ячейке) стольких единиц, сколько их в другом слагаемом (2-я ячейка). Поиск способа реализации ме-

Структура памяти и система команд МНР

Память: 1 2 3 4 ...

Код команды	Пояснения	Правила выполнения
Z (n) S (n) T (m, n,)	zero — нуль single — единичный transport — перенос	Обнуление ячейки с номером n Добавление к содержимому n-й ячейки единицы Копирование содержимого из m-й ячейки в n-ю
J (m, n, g)	jump — прыжок	Условный переход к команде с номером g при совпадении содержимого двух ячеек m и n. При несовпадении выполняется следующая в порядке написания команда
I (n) O (n)	input — ввод output — вывод	Ввод — запись числа в n-ю ячейку Вывод — чтение числа из n-й ячейки

тода — необходимость в счетчике добавляемых единиц (3-я ячейка). Организация цикла по счетчику: инициализация цикла — обнуление счетчика (команда 3), формирование условия выхода из него: при совпадении значений счетчика и второго слагаемого (команда 4), содержание тела цикла — добавление единицы в первое слагаемое и счетчик (команды 5 и 6), формирование команды безусловного перехода к началу цикла (команда 7), ввод исходных данных (команды 1 и 2) и вывод результата (команда 8). Поучительно «вручную» потестировать эту программу, взяв, например, числа 2 и 3 исходными данными и отмечая после выполнения очередной команды состояние ячеек памяти. Познавательно нарушить правила МНР, предъявив программе сложения числа —2, 3 и 2, —3. Можно поставить задачу на составление программы умножения двух натуральных чисел, которая приводит к вложенным циклам, нахождения наибольшего из двух чисел или др., преследуя те или иные дидактические цели. Но что надо сделать обязательно — показать избыточность команды копирования T (m, n), составив программу, использующую только команды Z, S и J и моделирующую работу команды T путем добавления к предварительно обнуленной ячейке n стольких единиц, сколько их в ячейке m. Избыточность команды T (m, n) позволяет предметно говорить о макрокомандах, подпрограммах и их компактных обозначениях, которые расширяют, пополняют исходную систему команд исполнителя более удобными, выполняющими более объемные операции из предметных областей, что подводит к программному обеспечению (библиотеки программ) и языкам программирования более высокого уровня.

В школьной информатике путь к языкам программирования лежит через надежно установленную опору — школьную алгоритмическую нотацию, так что достаточно «установить навигацию» от МНР до нее. Принципиальными шагами в этом направлении являются следующие:

1. Переход от нумерации ячеек к их именованию.

2. Обобщение операций Z, S и T в одну операцию присваивания, связывающую два операнда — получателя и отправителя информации (на этом уровне можно уже составлять не только блок-схемы, но и Бейсик-программы и моделировать работу МНР на реальном компьютере).

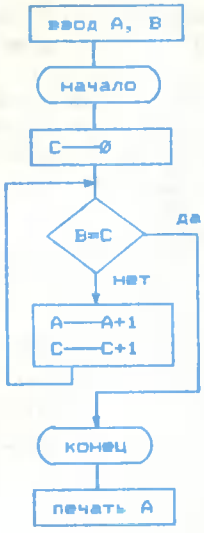
3. Установление, обоснование и оформление стандартных комбинаций управляющих структур: следование, ветвление, выбор, цикл, подпрограмма, функция.

4. Структуры данных (массивы) и расширение типов (целые, вещественные, литерные, логические и др.) обрабатываемых данных, включение специальных функциональных операций над ними (арифметические и логические операции, вычисление элементарных функций и т. д.).

Первые три шага проиллюстрированы на примере алгоритма сложения двух чисел (см. рисунок).

Другое направление изучения информатики исходя из МНР может быть начато с представления данных в ячейке памяти ЭВМ. Первой напрашивается классическая тема о системах счисления, далее — битовая структура ячейки, элементная база и поколения ЭВМ, кодирование информации и модели данных.

Уточняя вопрос о способах исполнения машиной отдельной команды, можно перейти к изучению архитектуры ЭВМ, микропроцессоров, машинных языков, ассемблеров, системному программированию. Добавляя к МНР различные внешние устройства (память, устройства ввода-вывода, связи с другими ЭВМ и т. д.) можно... Со- и противопоставляя МНР реальному компьютеру, можно двигаться в любых направлениях современной информатики, иметь возможность через МНР перейти к изучению других областей (см. схему). Теоретической базой такого подхода является тезис Чёрча — Тьюринга — Поста [1, 2], провозглашающий единство алгоритмического мира — эквивалентность любых алгоритмических моделей; возможность взаимного моделирования любой пары моделей с обеспечением совпадения результатов.

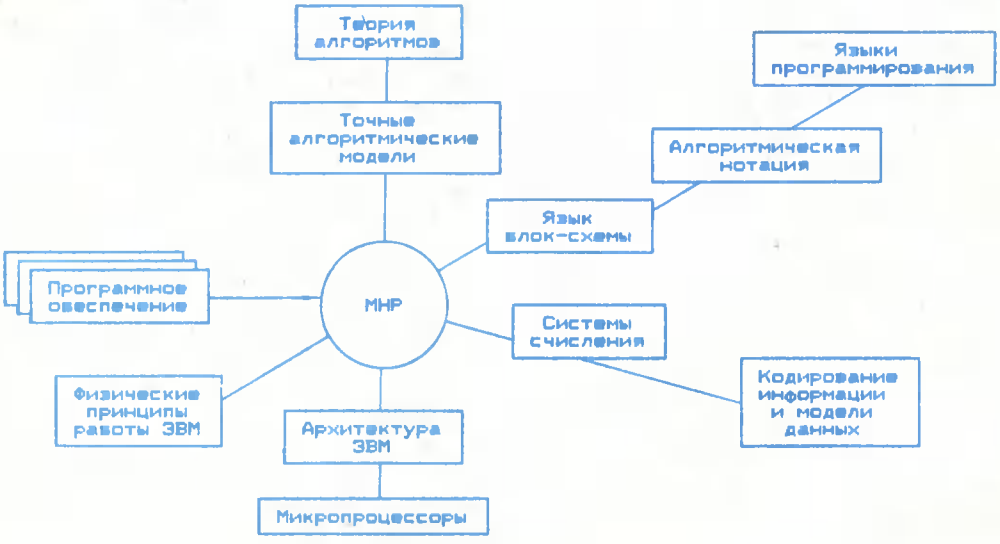



```
10 INPUT A,B
20 X=A
30 C=0
40 IF B=C GOTO 80
50 X=X+1
60 C=C+1
70 GOTO 40
80 PRINT X
90 END
```

```
алг нат сумма(нат A,B)
нач нат счетчик
   знач:=A
   счетчик:=0
   пока счетчик<>B
   нц
      знач:=знач+1
      счетчик:=счетчик+1
   кц
кон
```

Простота МНР позволяет изучать информатику в любых условиях (машинный и безмашинный варианты), в аудитории любого количественного и качественного состава (студенты, школьники младших классов, учителя средних учебных заведений, профессорско-преподавательский состав вузов, специалисты разного профиля на факультетах повышения квалификации), в рамках любых форм общения (в кружках, в коридоре или на прогулках по улице). Продолжительность процедуры знакомства с МНР легко регулируется аудиторией и занимает от нескольких минут до нескольких занятий.

Предлагаемый подход не столько выражает личные вкусы автора статьи, сколько касается фундамендальной проблемы рациональной организации знаний. Нынешний этап становления школьной информатики чрезмерно озабочен арифметической суммой знаний, сосредоточенным вниманием к программированию, упрощенным представлением последнего как одноэтапного процесса. Думается, однако, что усвоение знания человеком отличается от ввода информа-



ции в ЭВМ (как и от наполнения кувшина водой или зарядки аккумулятора), а программирование, если фиксировать на нем внимание, есть последовательность конверсий алгоритмических моделей от некоторой абстрактной (предметной) до заданной базовой.

Непосредственное отношение к проблеме обучения имеет привлекательная мысль А. Пуанкаре о том, что в своем индивидуальном умственном (как и физическом) развитии (онтогенез) человек повторяет историю своих предков, т. е человеческого общества (филогенез) [3, с. 11]. Действительно: куклы, машинки, пистолетики, «Му-му», закон Ома... — этапы социализации личности. С этой гуманитарной позиции, представляется, и следует формировать и укреплять процесс развивающего обучения: выявлять внутреннюю логику развития предмета изучения, выделять в ней концептуальные «точки роста» и «ветвления», очищать их от контекстуальных исторических случайностей и технических сложностей, определять логическое место каждой и выстраивать их во взаимообусловленную последовательность.

Литература

1. *Успенский В. А., Семенов А. Л.* Теория алгоритмов: основные открытия и приложения. М.: Наука, 1987.

2. *Катленд Н.* Вычислимость. Введение в теорию рекурсивных функций. М.: Мир, 1983.

3. *Лакатос И.* Доказательства и опровержения. М.: Наука, 1967.

4. Простое и сложное в программировании. М.: Наука, 1988.

5. *Вирт Н.* Систематическое программирование. М.: Мир, 1977.

6. *Вьюкова Н. И., Галатенко В. А., Ходулев А. Б.* Систематический подход к программированию. М.: Наука, 1988.

Вниманию читателей!

В ноябре 1990 г. в павильоне «Народное образование» ВДНХ СССР будет проходить выставка «Новые педагогические программные средства для средней школы».

На выставке предполагается показать лучшие программные продукты по общеобразовательным дисциплинам, курсу «Основы информатики и вычислительной техники», спецдисциплинам для профессиональной подготовки учащихся, игровые программы, а также программы для вузов.

В период работы выставки предусматривается проведение коммерческой работы, вам предложат коммерческого партнера для реализации и внедрения ваших разработок.

Предложения по участию в выставке должны быть представлены по адресу: **107078, Москва, ул. Садово-Черногрязская, д. 4 АСП «КУДИЦ». Тел. 207-00-69.**

АСП «КУДИЦ» предлагает организациям и авторским коллективам заключать договора на распространение разработанных для IBM совместимых ПЭВМ программно-педагогических средств.

В Центре компьютерного обучения при ИАЭ им. И. В. Курчатова СК «Полином» продолжает заключение договоров с авторами на коммерческое тиражирование педагогических программных средств по договору с ГКНО СССР.

Желающие приобрести ППС могут получить каталоги по почте, а также лично ознакомиться с работой имеющихся у нас программ.

Адрес: 123182 Москва, ул. Василевского, 9, кор. 1.

Телефон для справок: 193-72-34 с 18 до 20.

С. ГРИГОРЬЕВ
г. Йошкар-Ола

Программирование на Прологе-Д

Программа на Прологе представляет собой базу знаний и следующий за ней вопрос. Построению базы знаний посвящена данная статья. Здесь рассматриваются структура программы и некоторые приемы технологии программирования, связанные с использованием встроенных предикатов, рекурсии, графики, специфичные для Пролога-Д. Как и прежде во всех случаях, где это необходимо, будут оговариваться различия Пролога-Д MSX и Пролога-Д БК-0010.

Факты и правила. Особенностью языка Пролог-Д, отличающей его от других языков, используемых для работы на ЭВМ, является его применение не для программирования, но главным образом для описания данных и правил их обработки. Построение базы знаний на Прологе-Д означает выявление множества исследуемых объектов и связей между ними, совокупность которых описывает явление или процесс: иными словами, создание информационно-логической модели описываемого на языке Пролог-Д явления или процесса.

Первый шаг построения базы знаний состоит в выявлении объектов и соотношений между ними, отвечающих на вопрос «что дано?». Такую информацию целесообразно представлять в виде совокупности фактов. Классическим примером фактографии служит англо-русский словарь. Записанный средствами Пролога-Д, он выглядит так:

```
русангл(мама,mammy);
русангл(небо,sky);
русангл(солнце,sun);
русангл(мальчик,boy);
```

и т. д. Подобные отношения представляют собой грамматическую конструкцию Пролога-Д, называемую фактом. Факт задается в виде функционала: имя и совокупность аргументов. В данном примере «русангл» — это имя; оно определяет информацию, записываемую в факте. Русские и английские слова мама, mammy, небо, sky, солнце, sun, мальчик, boy представляют собой аргументы фактов, определяющих взаимно-однозначное соответствие между русскими и английскими словами. Необязательно, чтобы факт имел два аргумента. Например, факт

```
мужчина(Николай);
```

имеет один аргумент Николай, а факт

```
родился(Петров,Иван,10,сентябрь,1979);
```

имеет пять аргументов. Однако с точки зрения синтаксиса языка Пролог-Д необходим хотя бы один аргумент. Если факт в базе знаний имеет имя и не имеет аргументов, то система выдаст сообщение о синтаксической ошибке.

Приведенный пример, по сути дела, уже является базой знаний. Перед этой базой знаний можно ставить различные вопросы:

?русангл(y, x); — если необходимо

узнать все слова, хранящиеся в базе знаний;

?русангл(мама, x); — чтобы узнать, как по-английски «мама»;

?русангл(x, sky); — чтобы узнать, что значит слово sky.

Для описания всего множества информации достаточно фактов. Однако если можно задать некоторые связи и отношения между объектами, то удается сократить число фактов и тем самым сделать базу знаний более лаконичной. Связи и отношения между объектами задаются правилами. При построении правил выделяется совокупность отношений, отвечающих на вопрос «что известно?». Правило можно построить, пользуясь известным принципом разделения исходной задачи на более простые, которые тоже, в свою очередь, могут быть разделены. Этот процесс известен под названием декомпозиции задачи. Декомпозиция заканчивается в тот момент, когда отношения связывают зафиксированные в базе знаний объекты. Например, в задаче о построении родственных отношений можно определить следующие правила:

```
бабушка(x,y)<-мама(x,z),мама(z,y);
бабушка(x,y)<-мама(x,z),папа(z,y);
дедушка(x,y)<-папа(x,z),папа(z,y);
дедушка(x,y)<-папа(x,z),мама(z,y);
```

Левая часть правила называется головой; разделенные запятыми выражения в правой части — целями; последние образуют тело правила.

Процесс декомпозиции не обязательно однозначен. Даже простой пример о родственниках допускает и иную трактовку. Если ввести правило, определяющее понятие «родитель»

```
родитель(x,y)<-мама(x,y);
родитель(x,y)<-папа(x,y);,
```

то бабушку и дедушку можно определить проще:

```
бабушка(x,y)<-мама(x,z),родитель(z,y);
дедушка(x,y)<-папа(x,z),родитель(z,y);
```

Если к только что записанным правилам добавить несколько фактов, определяющих мам и пап, то получается база знаний «семья»:

```
мама(Саша,Петя);
папа(Сережа,Петя);
мама(Оля,Саша);
папа(Коля,Саша);
мама(Люда,Сережа);
папа(Петя,Сережа);
родитель(x,y)<-мама(x,y);
родитель(x,y)<-папа(x,y);,
бабушка(x,y)<-мама(x,z),родитель(z,y);
дедушка(x,y)<-папа(x,z),родитель(z,y);
```

В данном примере для определения понятия *родитель(x, y)* потребовалось более одного правила. По сути дела, здесь использовано недетерминированное ветвление, дающее альтернативное определение этого отношения и используемое системой после того, как было применено первое отношение. Следует подчеркнуть, что в определении участвуют оба правила. В общем случае число правил не ограничено.

Упражнения.

1. В электронике известно понятие «стандартный ряд номинальных значений сопротивлений»: 10, 11, 12, 13, 15, 16, 18, 20, 22, 24, 27, 30, 33, 36, 39, 43, 47, 51, 56, 62, 68, 75, 82, 91. Любое сопротивление, выпущенное промышленностью, имеет номинал, кратный элементам указанного ряда. Напишите базу знаний, в которой описывается этот ряд.

2. Опишите на языке Пролог-Д состав своей семьи.

3. Составьте базу знаний, описывающую республики, входящие в состав СССР.

4. Напишите на языке Пролог-Д таблицу умножения чисел от 1 до 10. Какое количество предложений требуется для записи этой базы знаний?

Арифметические и другие встроенные предикаты в Прологе-Д. Системы логического программирования, к числу которых относится и Пролог-Д, не предназначены для вычислений. Традиционный для Пролога-Д подход при выполнении арифметических действий дан в упражнении 4 из предыдущего раздела. Однако для определения таким образом всех математических действий памяти компьютера будет явно недостаточно. Поэтому традиционные действия, связанные с выполнением арифметических операций, осуществляются посредством специальных, так называемых встроенных, предикатов.

В системе Пролог-Д БК-0010 для выполнения арифметических действий

предусмотрен один встроенный арифметический предикат:

```
ВЫЧ(Apr1,Apr2,Apr3,Apr4)
```

В системе Пролог-Д MSX для выполнения арифметических действий предусмотрены два встроенных арифметических предиката:

```
УМНОЖЕНИЕ(Apr1,Apr2,Apr3,Apr4)
СЛОЖЕНИЕ(Apr1,Apr2,Apr3)
```

Встроенный предикат ВЫЧ (в MSX УМНОЖЕНИЕ) имеет четыре аргумента (целых; переменных, конкретизированных целыми; неконкретизированных переменных). ВЫЧ (в MSX УМНОЖЕНИЕ) обеспечивает реализацию формулы

```
Apr1*Apr2+Apr3=Apr4.
```


Предикаты предусматривают обратимость аргументов и полностью покрывают арифметические операции в области целых чисел, предусмотренных синтаксисом входного языка ($0 \leqslant$ ⟨число⟩ $\leqslant 65535$ «Электроника БК-0010» и $-32766 \leqslant$ ⟨число⟩ $\leqslant 32766$ в системе MSX). Если результат дробный, он округляется до целого. Оба предиката могут быть использованы только в качестве цели в предложении.

Предикат СЛОЖЕНИЕ (Арг1,Арг2, Арг3) обеспечивает реализацию формулы.

```
Apr1+Apr2=Apr3.
```

Следующая база знаний на языке Пролог-Д показывает, как можно описать любые арифметические операции в системе Пролог-Д БК-0010:

```
сложение(X,Y,Z)<-ВЫЧ(1,X,Y,Z);
вычитание(X,Y,Z)<-ВЫЧ(1,X,Z,Y);
умножение(X,Y,Z)<-ВЫЧ(X,Y,Ø,Z);
деление(X,Y,Z)<-ВЫЧ(Y,Z,Ø,X);.
```

Во всех четырех случаях X, Y — операнды операций, а Z — результат. Например, СЛОЖЕНИЕ(X, Y, Z) реализует арифметическую операцию Z=X+Y.

Аналогично в системе Пролог-Д MSX:

```
вычитание(X,Y,Z)<-УМНОЖЕНИЕ(1,X,Z,Y);
умнож(X,Y,Z)<-УМНОЖЕНИЕ(X,Y,Ø,Z);
деление(X,Y,Z)<-УМНОЖЕНИЕ(Y,Z,Ø,X);
```

Обратите внимание, вычитание можно определить в MSX и иначе:

```
вычитание(X,Y,Z)<-СЛОЖЕНИЕ(Y,Z,X);
```

Более подробное описание синтаксиса встроенных арифметических предикатов ВЫЧ, УМНОЖЕНИЕ, СЛОЖЕНИЕ приведено в технических описаниях трансляторов.

Пример 1. На Прологе-Д опишем вычисление площади прямоугольника, имеющего стороны длиной *a* и *b*:

$$S=ab.$$

Соответствующий предикат должен иметь три аргумента — длины сторон и величину площади. Имя предиката должно отражать его назначение, этому критерию удовлетворит имя *площадь*.

В системе Пролог-Д БК-0010:

```
умножение(X,Y,Z)<-ВЫЧ(X,Y,Ø,Z);
площадь(a,b,S)<-умножение(a,b,S);.
```

Первый предикат *умножение* потребовалось определить для наглядности записи. Необходимо отметить, что предикат *площадь* обратим, это означает, что, пользуясь этим описанием, можно вычислить не только площадь по заданным сторонам, но и любую (одну) сторону по другой стороне и площади. Перед базой знаний можно поставить вопрос:

```
?площадь(1Ø,2Ø,S);
```

Ответ системы Пролог-Д:

```
S=2ØØ
?площадь(a,2Ø,1ØØ);
```

Ответ системы Пролог-Д:

```
a=5
```

В системе Пролог-Д MSX:

```
умнож(X,Y,Z)<-УМНОЖЕНИЕ(X,Y,Ø,Z);
площадь(a,b,S)<-умнож(a,b,S);.
```

Пример 2. На Прологе-Д необходимо описать вычисление объема параллелепипеда высотой *h*, в основании которого прямоугольник, имеющий стороны длиной *a* и *b*:

$$V=abh.$$

Соответствующий предикат должен иметь четыре аргумента — длины сторон *a*, *b*, высоту *h* и объем V.

В системе Пролог-Д БК-0010:

```
умножение(X,Y,Z)<-ВЫЧ(X,Y,Ø,Z);
объем(a,b,h,V)<-умножение(a,b,S),
                умножение(S,h,V);
```

Как и прежде, предикат *объем* обратим, это означает, что, используя это описа-

ние, можно вычислить не только объем по заданным сторонам и высоте, но и любую (одну) сторону или высоту по высоте, стороне и объему.

Можно иначе записать объем, если воспользоваться формулой

$$V=Sz.$$

Эту базу знаний предлагается написать самостоятельно.

Перед данной базой знаний можно поставить вопрос:

```
?объем(10,20,5,V);
```

Ответ системы Пролог-Д:

```
V=200
```

В системе Пролог-Д MSX:

```
умнож(X,Y,Z)<-УМНОЖЕНИЕ(X,Y,0,Z);
объем(a,b,h,V)<-умнож(a,b,S),
                 умнож(S,h,V);
```

Наряду с арифметическим предикатом в системе БК-0010 существуют два предиката БОЛ и НЕ.

Встроенный предикат БОЛ(Арг1,Арг2) предназначен для сравнения двух целых констант или переменных. Он имеет два аргумента (целых или переменных, конкретизированных целыми). Оба аргумента к моменту выполнения должны быть определены. Если эти требования не выполнены, то появится сообщение об ошибке: «Функция не может быть выполнена (ошибка 34)». Предикат выполнен, если Арг1>Арг2, иначе не выполнен.

Несмотря на то что предикат БОЛ один, его достаточно для описания всех возможных предикатов для сравнения числовой информации: равенство — *равно*; меньше — *меньше*; меньше и равно — *мир* и т. д. Это показывает база знаний, приведенная ниже.

```
равно(X,X);
меньше(X,Y)<-БОЛ(Y,X);
мир(X,Y)<-НЕ(БОЛ(X,Y));
```

В последнем предложении использован встроенный предикат НЕ, его синтаксис:

```
НЕ(Арг1);
```

Он имеет один аргумент, который обязательно должен быть предикатом. Предикат НЕ выполнен тогда и только тогда, когда предикат-аргумент не выполнен.

А теперь несложный пример, иллюстрирующий применение БОЛ и НЕ.

Пример 3. Опишите на языке Пролог-Д вычисление функции Хевисайда, определяемую формулой:

$$h(x)=\begin{cases}0, & \text{если } x\leqslant 0;\\ 1, & \text{если } x>0.\end{cases}$$

База знаний должна содержать описание предиката «меньше и равно», который выше уже был описан; предикат, выполняющийся при вычислении функции Хевисайда, будет называться *хевисайд*. Этот предикат будет иметь два аргумента: первый — аргумент функции, второй — ее значение. Предикат *хевисайд* определяется через два альтернативных описания для обоих значений *x*.

```
мир(X,Y)<-НЕ(БОЛ(X,Y));
хевисайд(X,0)<-мир(X,0);
хевисайд(X,1)<-БОЛ(X,0);
```

Перед этой базой знаний можно поставить различные вопросы.

```
?хевисайд(20,X);
```

Ответ системы Пролог-Д:

```
X=1
```

В системе Пролог-Д MSX существует полный набор предикатов сравнения, их имена БОЛЬШЕ, МЕНЬШЕ, РАВНО.

Еще один, последний, встроенный предикат — «отсечение», предназначенный для управления логическим выводом. Этот предикат потребуется для решения следующих проблем:

ограничения количества найденных решений;

нахождения некоторого особенного решения задачи;

ограничения объема поиска с целью повышения эффективности работы системы.

Предикат «отсечение» обозначается восклицательным знаком !. Это традиционное обозначение отсечения в системах логического программирования. Если данный предикат использовать в качестве цели в предложении, то полученный при этом эффект можно проиллюстрировать дверью, через которую можно пройти только слева направо, но нельзя вернуться назад через эту дверь. Роль двери выполняет символ !. Как известно, система Пролог-Д будет пытаться выполнять цели предложения в порядке просмотра слева направо начи-

ная от символа ←, от первой до последней цели. Если какая-либо цель оказывается невыполненной, то осуществляется возврат и делается попытка найти альтернативные решения. Отсечение ограничивает возможность поиска альтернатив с того момента, как была просмотрена цель, обозначенная символом !. Например, если не выполнены цели А,Б,В, возврат для нахождения альтернативных решений в предложении

```
пример<-А,Б,В,!,Г,Д,Е;
```

возможен, а если не выполнены цели Г,Д или Е, то уже нет. Рубикон при движении слева направо перейден.

Необходимо отметить важность этого предиката, особенно при описании задач, допускающих множественные решения.

Иллюстрация предиката *отсечение* на примере базы знаний *мама*. Действительно, у человека не может быть две матери, поэтому, определив для данного человека имя матери, необходимо прекратить дальнейшие поиски.

```
мама(Наташа,Петя)<-!;
мама(Наташа,Ваня)<-!;
мама(Оля,Лена)<-!;
мама(Катя,Даша)<-!;
мама(Люда,Сережа)<-!;
мама(Лена,Костя)<-!;
```

Перед базой знаний может быть поставлен вопрос

```
?мама(x,Петя);
```

Ответ системы Пролог-Д:

```
      x=Наташа
ДРУГИХ РЕШЕНИЙ НЕТ
```

После отыскания первого решения поиск альтернатив не производится. Что будет, если убрать во всех предложениях предикаты отсечения?

Упражнения.

1. Опишите на языке Пролог-Д вычисление площадей геометрических фигур: трапеции, треугольника, параллелограмма.

2. Опишите вычисление площади круга и длины окружности. Какова точность вычислений этих величин? Можно ли вычислить радиус круга по длине окружности?

3. На языке Пролог-Д напишите базу знаний, в которой определяется функция, заданная соотношением:

$$F(x) = \begin{cases} x^2, & \text{если } x<1, \\ x+1, & \text{если } -1 \leqslant x<1, \\ x^2, & \text{если } x>1. \end{cases}$$

4. Какие сложности могут возникнуть в базе знаний о мамах, если у двух мам дети будут тезками?

5. Предположим, что дана программа на Прологе-Д:

```
ff(xx);
ff(x)<-pp(x),ff(x);.
```

Каким должен быть предикат *pp(x)*, чтобы система нашла одно решение; бесконечно много решений?

Рекурсия. Существует огромное количество задач, в которых отношения между объектами можно определить, только используя сами определяемые соотношения. При этом получаются правила, называемые рекурсивными. Применение рекурсии для описания задач при работе с системами логического программирования — широко распространенный прием.

Рекурсия будет проиллюстрирована несколькими примерами построения программ, как вычислительных, так и логических.

Первым примером будет пример вычисления наибольшего общего делителя (НОД) двух чисел. Предикат, который выполняется, если найден НОД двух данных чисел, будет иметь имя *нод* и три аргумента: числа *a*, *b* и значение НОД — *c*. Для описания вычисления НОД используются следующие соображения:

если $a=b$, то $c=a=b$;

если $a>b$, то необходимо вычислить НОД для чисел b и $a-b$;

если $b>a$, то необходимо вычислить НОД для чисел a и $b-a$.

Эти три утверждения естественным образом могут быть записаны на Прологе-Д в системе БК-0010.

```
нод(a,a,a);
нод(a,b,c)<-БОЛ(a,b),вычитание(a,b,d),
                              нод(b,d,c);
нод(a,b,c)<-БОЛ(b,a),вычитание(b,a,d),
                              нод(a,d,c);
вычитание(X,Y,Z)<-ВЫЧ(1,Y,Z,X);
```

Если перед этой базой знаний поставить вопрос

```
?нод(10,15,x);
```

то система Пролог-Д ответит:

```
    x=5
ДРУГИХ РЕШЕНИЙ НЕТ
```

Предикат *нод* оказывается обратимым.

В системе Пролог-Д MSX последняя строка базы знаний запишется иначе:

```
вычитание(X,Y,Z)<-СЛОЖЕНИЕ(Y,Z,X);
```

а предикат БОЛ необходимо заменить на БОЛЬШЕ. Других изменений нет.

В качестве второго примера рассмотрим задачу о вычислении элементов последовательности

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, ..., известной как последовательность Фибоначчи. Каждый ее элемент определяется следующими правилами:

$f_0=0$,

$f_1=1$,

$f_n=f_{n-1}+f_{n-2}$.

Первая формула утверждает, что значение нулевого элемента последовательности равно нулю. Это можно записать в виде факта

```
Фб(0,0);
```

Вторая утверждает, что первый элемент равен 1. На Прологе-Д это можно записать так:

```
Фб(1,1);
```

Третья формула представляет собой запись рекурсивного соотношения, переписываемого так:

```
Фб(N,X)<-БОЛ(N,1),вы(N,1,M),вы(N,2,K),
                Фб(M,Y),Фб(K,Z),сл(Y,Z,X);
```

вы и *сл* — имена предикатов *вычитание* и *сложение*, определенных с помощью правил

```
сл(X,Y,Z)<-ВЫЧ(1,X,Y,Z);
вы(X,Y,Z)<-ВЫЧ(1,Y,Z,X);
```

в системе БК-0010. Использование таких обозначений объясняется тем, что длина одной строки в редакторе системы Пролог-Д БК-0010 не должна превышать 64 символов.

Данные предложения представляют собой базу данных на языке Пролог-Д, позволяющую вычислять значения элементов последовательности. На вопрос

```
?Фб(8,X);
```

система Пролог-Д ответит:

```
    x=21
ДРУГИХ РЕШЕНИЙ НЕТ
```

Поиск числа Фибоначчи с порядковым номером более 8 невозможен в БК-0010 из-за ограниченной памяти, поэтому рекомендуется первым предложением базы знаний написать

```
Фб(N,Y)<-БОЛ(N,8),!;
```

Необходимо сказать, что такой путь решения данной задачи не самый лучший. Для нахождения N+1 числа Фибоначчи требуется 2(N+1)—1 рекурсивное обращение. Однако этого можно избежать, если перейти к другой базе знаний, в которой предикат с именем *Фиб* определен как имеющий три аргумента, включающий в себя в качестве аргумента значения N-го и N—1-го элементов последовательности. Запишем эту базу знаний.

1. Нулевой элемент последовательности равен нулю, а «минус первый» не определен.

```
Фиб(0,x,0);
```

В данном случае значение *x* может быть любым.

2. Первый элемент последовательности равен 1, а нулевой — 0.

```
Фиб(1,0,1);
```

3.

```
Фиб(N,F,f)<-БОЛ(N,1),вы(N,1,M),
           ---------
                Фиб(M,Ф,F),сл(Ф,F,f);.
```

Обращение к этой базе знаний будет иметь вид:

```
?Фиб(10,F,f);
```

Ответ системы Пролог-Д:

```
    F=55,
    f=34
ДРУГИХ РЕШЕНИЙ НЕТ
```

При работе с этой базой знаний для вычисления N-го числа Фибоначчи необходимо всегда лишь N рекурсивных обращений.

Для системы Пролог-Д характерна особенность, проявляющаяся при работе с рекурсивными программами. В общем случае порядок предложений в базе знаний не имеет значения. Однако в нижеследующем примере это не так.

```
родитель(X)<- родитель(Y),отец(Y,Z);
родитель(Коля);
отец(Коля,Петя);
родитель(Петя);.
```

В первом предложении голова имеет то же имя, что и одна из целей — «родитель». В процессе поиска ответа в этой базе знаний будет применено правило «предложение, стоящее первым, будет и применено первым», известное как принцип поиска в глубину. Это приведет к тому, что система будет обращаться только к первому предложению базы знаний и ответ на вопрос

```
?родитель(Петя);
```

не будет найден никогда. Вместе с тем небольшое изменение базы знаний, перестановка двух предложений, приводит к удачному поиску решения:


```
родитель(Коля);
родитель(X)<-родитель(Y), отец(Y,X);
отец(Коля,Петя);
?родитель(Петя);.
```

Неограниченно-повторное обращение к предложению может быть и более замаскированным, как это получается в примере:

```
выше(A,B)<-ниже(B,A);
ниже(B,A)<-выше(A,B);
выше(Коля,Петя);
?ниже(Петя,Коля);.
```

Однако если третье предложение стоит на первом месте, то повторного обращения не произойдет и ответ будет найден. Такая ситуация называется петлей.

При вычислении элементов последовательности Фибоначчи может появляться бесконечная петля при исполнении программы. В самом деле, если вопрос имеет вид

```
?Фиб(Ø,x,y);
```

то первый возможный результат

```
x=Ø,
y=1
```

Далее в попытке отыскать следующее решение возникает бесконечная петля, так как будет отыскиваться

```
Фиб(-1,x,y), Фиб(-2,...),... .
```

Для контроля за подобной ситуацией необходима модификация базы знаний. Первые два предложения должны быть записаны в виде

```
Фиб(Ø,x,1)<-!;
Фиб(1,1,1)<-!;,
```

тогда при поиске альтернативного решения после получения ответа на вопрос:

```
?Фиб(Ø,x,y);
x=Ø,
y=1
```

будет получен результат: БОЛЬШЕ РЕШЕНИЙ НЕТ. Данный пример иллюстрирует первое возможное использование предиката «отсечение».

Еще одно, чисто эстетическое предложение. База знаний на Прологе-Д будет выглядеть лучше, если предложения с одинаковыми именами расположены в одном месте. Для сравнения приводятся две базы знаний:

```
1.
мама(Таня,Надя);
бабушка(X,Y)<-мама(X,Z),
мама(Z,Y);
мама(Надя,Катя);

2.
мама(Таня,Надя);
мама(Надя,Катя);
бабушка(X,Y)<-мама(X,Z),
мама(Z,Y);.
```

Упражнения.

1. Написать базу знаний, описывающую вычисление факториала.
2. Написать базу знаний, описывающую вычисление суммы чисел натурального ряда.
3. Написать базу знаний, описывающую вычисление суммы квадратов чисел натурального ряда.
4. Описать вычисление наименьшего общего кратного.
5. Написать базу знаний, описывающую известную притчу «У попа была собака, он ее убил. Она съела кусок мяса, он ее убил, на могиле написал...» и т. д. Как сделать, чтобы эта база знаний не содержала петли?
6. Написать базу знаний, описывающую сказку о репке, которую тянут-потянут, а вытянуть не могут.

А. ЛЕПИН, Л. ЛЕПИН, Т. ТРЕЙКАЛЕ

«Редактор экрана»— инструмент разработчика

Опыт показывает, что ВТ используется в образовании нерационально: в школьных компьютерных классах, как правило, проводятся только уроки по курсу ОИВТ, в вузах же на персональных ЭВМ обучают программированию, и в основном студентов технических специальностей. В то же время реальные возможности ВТ позволяют применять ее при обучении практически по всем дисциплинам; препятствует этому почти полное отсутствие качественных обучающих программ. Первоначальные надежды на то, что по мере насыщения учебных заведений ЭВМ банк таких прогрмм будет постоянно пополняться, не оправдались. Причиной этого является как низкая программистская квалификация педагогов-практиков, так и методическая неподготовленность профессиональных программистов. Первые плохо представляют себе архитектурные возможности конкретных ПЭВМ, что отражается на написанных ими сценариях обучающих программ и усложняет реализацию последних. Программисты же не знакомы с требованиями педагогической методики, и разработанные ими фрагменты отдельных курсов не могут служить основой для создания АОС.

Решение проблемы видится в организации разнообразных способов сотрудничества программистов и педагогов с привлечением и специалистов других профилей. Один из возможных путей: профессиональные программисты создают инструментальное матобеспечение, приспособленное к архитектуре конкретной ПЭВМ и доступное для использования после непродолжительного обучения. При работе с ним в диалоговом режиме преподаватели будут создавать отдельные блоки будущих программ по заранее разработанному сценарию и описывать связи между ними. Пользуясь этими сценариями и библиотекой стандартных подпрограмм, программисты скомпонуют из блоков законченную программу. Таким образом, создание методически качественных и хорошо оформленных обучающих программ можно будет поставить на поток, обеспечив к тому же их низкую себестоимость.

Иллюстрации, помещенные в этой статье и на четвертой странице обложки, созданы с помощью «Редактора экрана».

Назначение «Редактора экрана»

Часто употребляемыми и наиболее доступными для автоматизации являются контролирующие программы, состоящие из тестов с ответами выборочного типа. Несмотря на ограниченную эффективность по сравнению с тестами, требующими описательного или результативного ответа, такие программы выигрывают в легкости подсчета результатов. С программистской точки зрения их создание может быть почти полностью автоматизировано при наличии соответствующего математического обеспечения. Одним из шагов к разработке такого рода матобеспечения стало создание программы «Редактор экрана», позволяющей упростить процесс программирования путем автоматизации процесса создания некоторых блоков программ.

Акцент в «Редакторе экрана» сделан на эстетическом оформлении визуальной информации и на максимальном упрощении процедуры управления программами.

«Редактор экрана» выполняет такие функции, как задание визуальной информации (текстовой и графической) на экране дисплея, задание управления программой, создание различных шрифтов и использование национальных алфавитов, облегчает перевод программ на другие естественные языки. Состоит он из шести подпрограмм, результатом работы которых является программный модуль на Бейсике. Последний в процессе исполнения обеспечивает вывод на экран рисунков, текста, анализ положения курсора и т. п.

«Редактор экрана» предназначен для работы на ПЭВМ «Ямаха» MSX-2. Он содержит 14 файлов общим объемом около 150К байт (в том числе файлы, содержащие инструкцию по его применению). Программы, написанные на Бейсике, занимают 83К байт, написанные в двоичных кодах — 2,5К байт.

Управление программами. Одним из важных вопросов, решаемых при создании обучающих программ, является выбор принципов управления программой. Следует учитывать, что эти программы должны быть доступны пользователям с разным уровнем навыков работы на персональных ЭВМ, а значит, максимально просты в обращении. В таких случаях управление осуществляется пользователем путем выбора очередного шага

из нескольких возможных, перечисленных на экране. Каждой из этих возможностей соответствует определенная область на экране дисплея, так называемая область выбора. Попадание курсора (его передвижение осуществляется с помощью управляющих клавиш клавиатуры) в одну из таких областей и последующее нажатие клавиши **ПРОБЕЛ** определяют, по какой ветви будет передано управление программой. Вместо клавиатуры можно использовать манипулятор «мышь», более точно определяющий движение курсора. Практика показывает, что именно «мышь» предпочитают пользователи, не имеющие опыта работы на ПЭВМ.

Управление осуществляют входящие в матобеспечение двоичные программы, а задание областей выбора автоматизирует одна из подпрограмм «Редактора экрана». Это задание производится визуально на экране дисплея после вывода на него соответствующего изображения (оно может создаваться с помощью этого же «Редактора экрана»). С помощью управляющих клавиш отмечаются желательные границы областей выбора, цвет и внешний вид курсора в каждой из них, границы общей области перемещения курсора, а также место расположения курсора при появлении данного экрана в программе. Полученный в результате модуль состоит из блока данных и операторов считывания и автоматически записывается на диск в виде файла.

Организация диалогового режима. Немалую часть времени при создании обучающих программ занимает организация диалогового режима их работы, т. е. реализация ввода информации и вывода ее на экран дисплея. Эту проблему в значительной мере решают три подпрограммы «Редактора экрана», позволяющие создавать новые экраны, редактировать уже имеющиеся, а также резервировать место для переменных величин. Нужное изображение создается (редактируется) непосредственно на экране дисплея, а автоматически генерируемые при этом операторы Бейсика, реализующие это изображение, группируются в отдельный файл, который и записывается на диск. «Редактор экрана» допускает задание как текстовой информации (состоящей из любых символов, имеющихся на клавиатуре), так и графической. В последнем случае он выполняет такие операторы Бейсика, как LINE, PSET, PAINT, CIRCLE с произвольными параметрами (первые два могут выполняться в цикле).

Вывод текстовой информации можно разнообразить, применяя следующие возможности «Редактора экрана»: изменение цвета символов, изменение вдвое их ширины и высоты, печать вразрядку, печать символов с оттенением (т. е. в два цвета), выбор рисунка шрифта. «Редактор экрана» использует четыре таблицы символов, включающие 256 символов каждая. В матобеспечение включена программа, позволяющая менять содержание таблиц и поэлементно, и целиком, вводить и редактировать новые символы. Среди символов могут присутствовать и буквы национальных алфавитов (в настоящем варианте — латышского), что дает возможность использования в обучающих программах любого языка.

Одновременно «Редактором экрана» могут применяться 16 цветов, но этим его возможности не исчерпываются. Матобеспечение позволяет использовать четыре палитры по 16 цветов каждая (за основную по умолчанию принимается стандартная палитра «Ямахи»).

Простота обращения с «Редактором экрана», возможность составления изображения непосредственно на экране позволяют привлечь к этой работе специалистов-дизайнеров, психологов, что, несомненно, повысит качество эстетического оформления и психологического воздействия обучающих программ.

Структура создаваемых программ. Как уже сказано, с помощью «Редактора экрана» можно создавать оригинальные обучающие программы на любом естественном языке (русском, латышском, английском и т. д.) либо переводить уже созданные с одного языка на другой. Для облегчения последней процедуры, выполняющие создание очередного экрана (именно они обычно подлежат переделке при необходимости перевода), группируются в виде подпрограмм, легко поддающихся замене путем присоединения нужных файлов. Такие файлы компонуются с помощью одной из подпрограмм «Редактора экрана» из файлов, созданных другими его подпрограммами. Такая структура результирующих программ упрощает процесс их создания и дает возможность для его дальнейшей автоматизации.

Применение «Редактора экрана». За время практического использования «Редактора экрана» с его помощью были разработаны комплексы обучающих программ «Логика» и «Теория множеств», применяемые в настоящее время при преподавании курса математического анализа студентам-первокурсникам физико-математического факультета ЛГУ им. П. Стучки. Используется он и в текущей работе по созданию инструментального матобеспечения для дальнейшей автоматизации процесса разработки обучающихся программ.

Рассмотрим подробнее применение «Ре-

дактора экрана» на примере создания комплекса «Логика». Комплекс состоит из программы загрузки, программы-меню, семи обучающих и одной контролирующей программ, каждая из которых в процессе своей работы выводит на монитор от трех до десяти экранов. По способу создания и выполняемым функциям эти экраны можно разделить на два типа: информативные и экраны с возможностью выбора.

Информативные экраны сменяют друг друга в заранее определенной последовательности после очередного нажатия обучаемым клавиши **ПРОБЕЛ.** В комплексе «Логика» они используются для создания заставки, вывода на дисплей теоретического материала и инструкции пользователю. Как правило, такие экраны задаются в программе в явном виде и не содержат переменных величин. В этом случае для их создания применяются две подпрограммы «Редактора экрана», реализующие создание нужного изображения и вывод его на экран (поэлементный, построчный, путем выплывания снизу и т. д.). В зависимости от сложности изображения создание файла информативного экрана может занимать от пяти минут до получаса.

Экраны с возможностью выбора, как видно из их названия, реализуют разветвленную структуру и сменяются на дисплее либо частично изменяются после выбора пользователем нужного ветвления с помощью управляющих клавиш и клавиши **ПРОБЕЛ.** Комплекс «Логика» выводит на монитор по этим принципам различные оглавления, некоторые страницы инструкции, а также все тексты. В общем случае разработка одного экрана с возможностью выбора требует использования четырех подпрограмм «Редактора экрана»: по сравнению с информативным экраном здесь необходимо еще резервирование места для переменных величин (например, в тестах с постоянной формулировкой вопроса и изменяемой частью задания) и определение областей выбора. В среднем создание одного файла экрана с возможностью выбора занимает 20—30 мин.

Следует также отметить, что при разработке «Логики» было существенно сохранение композиционного единства комплекса, а следовательно, единообразия программ, в том числе и с оформительской точки зрения. Поэтому создание новых экранов обоих типов превращалось фактически в редактирование уже имеющихся экранов, что позволяло экономить время и упрощать работу.

Хотя первоначально «Редактор экрана» создавался для интенсификации работ по программированию курса математического анализа, его возможности гораздо шире. Практически он может применяться для создания обучающих программ произвольной тематики как по вузовским, так и по школьным курсам и после непродолжительного обучения доступен любому преподавателю.

Адрес для справок: 226250, Рига, ГСП, бульвар Райниса, 29, ВЦ при ЛГУ им. П. Стучки, отдел обыкновенных дифференциальных уравнений.

В. ТИЩЕНКО

Простейший компьютер

Компьютеров в СССР, увы, не избыток. И во многих школьных кружках радиоэлектроники и вычислительной техники, кружках станций юных техников да и просто дома энтузиасты пытаются собрать компьютер своими руками. У них обычно возникают две проблемы. Во-первых, трудно в условиях нашего дефицита приобрести несколько десятков микросхем; во-вторых, очень тяжело бывает разобраться в необычной, программной, работе компьютера (освоить язык программирования), без чего невозможно наладить собранную конструкцию и эффективно работать с ней.

Чтобы решить упомянутые проблемы, в кружке «Компьютер» Киевской городской станции юных техников была разработана простейшая схема, для сборки которой нужны не десятки, а лишь две основные микросхемы. Такой компьютер облегчает изучение принципа его работы и позволяет успешно освоить азы программирования.

Как видно из функциональной схемы (рис. 1), простейший компьютер состоит из двух основных модулей: микропроцессорного и памяти. Ввод программы в память, управление режимом работы и выдача результата производятся при помощи модулей ввода, вывода и управления. Основные и дополнительные (указанные на рис. 1 пунктиром) модули соединяются друг с другом при помощи общей шины.

Общая шина представляет собой жгут проводов, состоящий из 16 проводов адресной шины, 8 — шины данных, 12 — шины управления, 4 — шины питания и 5 резервных.

Соединение общей шины с модулями компьютера выполняется при помощи многоконтактных разъемов. На модулях устанавливаются штыревые части разъема, а на общей шине — гнездовые.

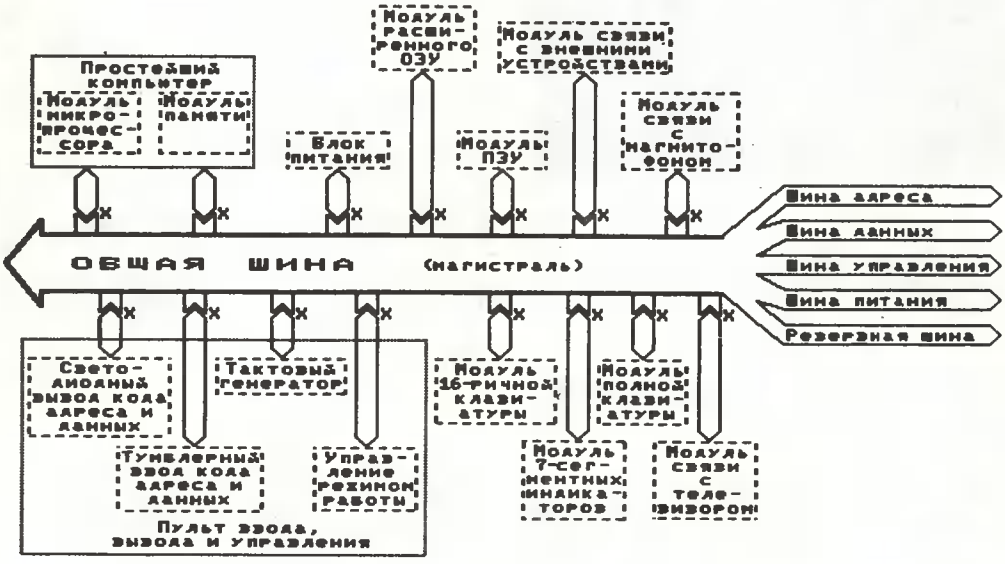
Для соединения использованы разъемы ГРПМ 1—45, имеющие 45 контактов. Допускается применение разъемов и другого типа, имеющих не менее 40 контактов.

Расположение и назначение контактов ГРПМ 1—45 показано на рис. 2.

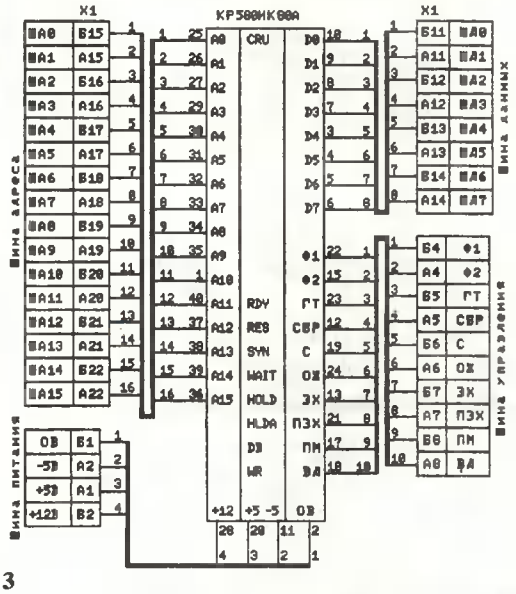
В качестве микропроцессорного модуля используется наиболее популярный и наименее дефицитный микропроцессор типа КР580ИК80А. В Киеве на «черном» радиолюбительском рынке КР580ИК80А продается по цене от трех до пяти рублей. Примерно столько же он стоит на рынках и других крупных городов. (Можно надеяться, что к дню опубликования этой статьи КР580ИК80А появится и в системе «Посылторга».) Соединение выводов КР580ИК80А с контактами разъема ГРПМ 1—45 показано на рис. 3.

В модуле памяти использована микросхема типа К537РУ10, так как она потребляет очень малый ток, что позволяет, добавив два элемента питания от электронных часов,

1



2

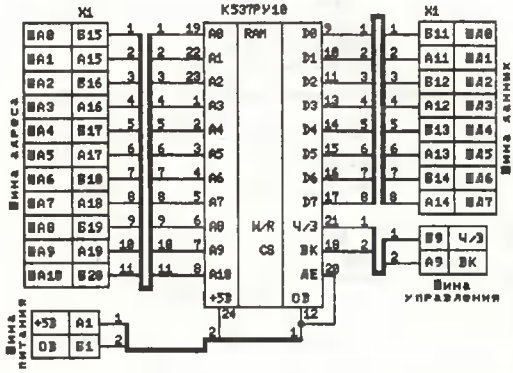


3

получить энергонезависимое запоминающее устройство. Вместо К537РУ10 можно использовать и другие микросхемы памяти, например К541РУ2.

Соединение выводов К537РУ10 с контактами разъема ГРПМ 1—45 показано на рис. 4.

4



Модули ввода, вывода и управления состоят из светодиодного вывода кода адреса и данных; тумблерного ввода кода адреса и данных; тактового генератора; узла управления режимом работы (рис. 5).

Код адреса и данных можно узнать при помощи светодиодов VD1—VD24, подключенных к проводам шин адреса и данных. Каждый светодиод сообщает своим свечением о том, что на проводе, к которому он подключен, установлен уровень логической единицы. При отсутствии светодиодов АЛ307Б их можно заменить светодиодами другого типа или лампочками от карманного фонаря.

Для уменьшения нагрузки на шины светодиоды (или лампочки) включаются через транзисторные усилители (VT1—VT24).

Для упрощения схемы подключение тумблеров SA1—SA24 к шинам адреса и данных (на время программирования памяти) производится вручную с помощью разъемов X2 и X3. Выполнять эту операцию следует внимательно и осторожно и только после того, как появится разрешающий сигнал ПЗХ микропроцессора (о чем сигнализирует светодиод VD25). Подключение тумблеров к шинам при отсутствии сигнала ПЗХ может привести к выходу из строя микропроцессора. Поэтому ручной способ подключения тумблеров SA1—SA24 следует признать временным, планируя в дальнейшем заменить разъемы X2 и X3 управляемыми буферными усилителями с тремя состояниями (типа К155ЛП8, К589АП16, К580ИК86 и др.), которые управляются сигналом ПЗХ и автоматически подключают или отключают тумблеры от шин при программировании или работе микропроцессора.

В случае отсутствия микросхем с тремя состояниями можно использовать широко распространенную микросхему с открытым коллектором К155ЛА8 (или подобную). На входе К155ЛА8 сигнал инвертируется, поэтому положение тумблеров нужно изменить на противоположное: для набора логической единицы (высокого уровня) их следует устанавливать в нижнее положение, а для набора логического нуля — в верхнее.

Примеры включения микросхем в разрыв разъема данных X3 показаны на рис. 6. Аналогично выполняется включение и для разъема адреса X2, только число микросхем увеличивается вдвое.

Тактовый генератор выполнен на микросхеме типа К580ГФ24. Его колебания стабилированы кварцем Z.

Таким образом, пульт управления изготавливается во много раз быстрее при параллельной творческой работе нескольких кружковцев.

В случае отсутствия кварца Z он может

быть заменен небольшой емкостью в 10—15 пФ же колебательным контуром, настроенным на частоту в 9 МГц.

Микросхему тактового генератора К580ГФ24 можно заменить набором обычных логических микросхем так, как это указано на рис. 7.

Цифры в скобках рядом со стрелками обозначают номера выводов микросхемы К580ГФ24, которые им соответствуют.

Узел управления — наиболее сложная часть схемы на рис. 5. Здесь элементы DD3 и DD4 совместно с кнопками SA29 и SA30 определяют режим чтения или записи при программировании памяти.

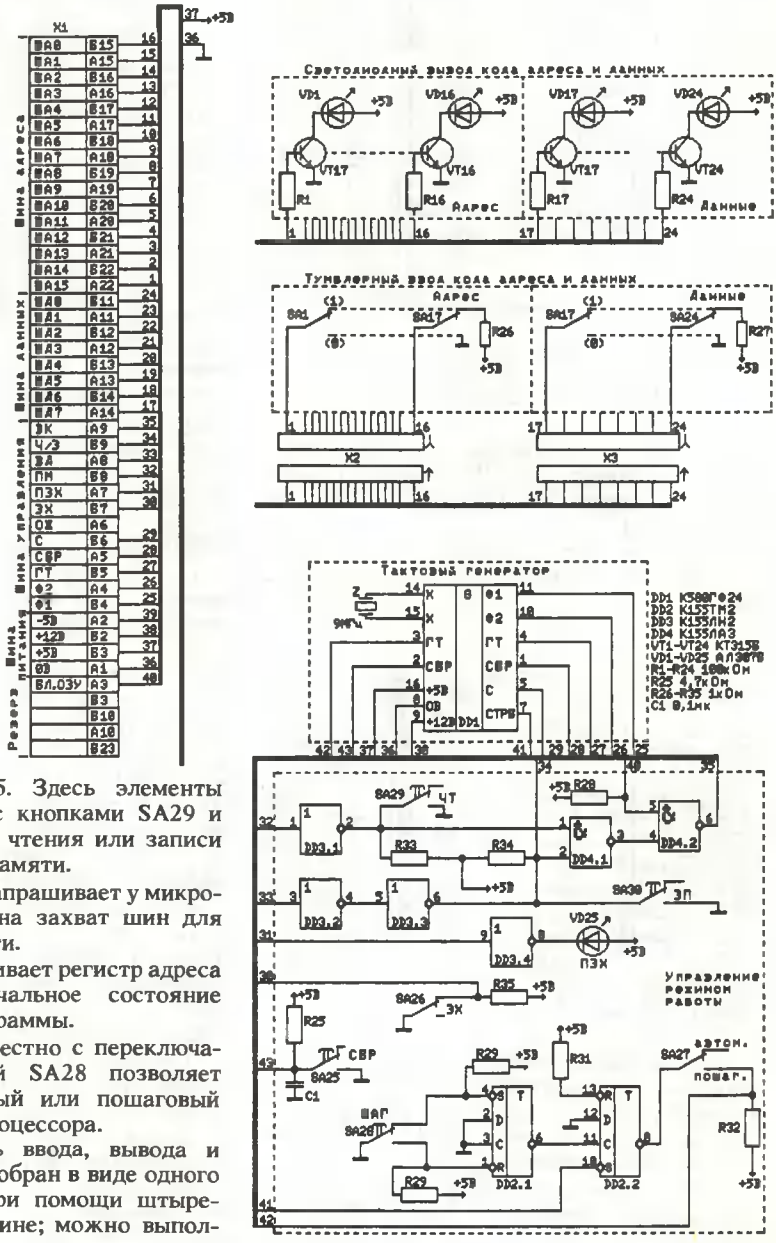
Переключатель SA26 запрашивает у микропроцессора разрешение на захват шин для программирования памяти.

Кнопка SA25 устанавливает регистр адреса микропроцессора в начальное состояние перед выполнением программы.

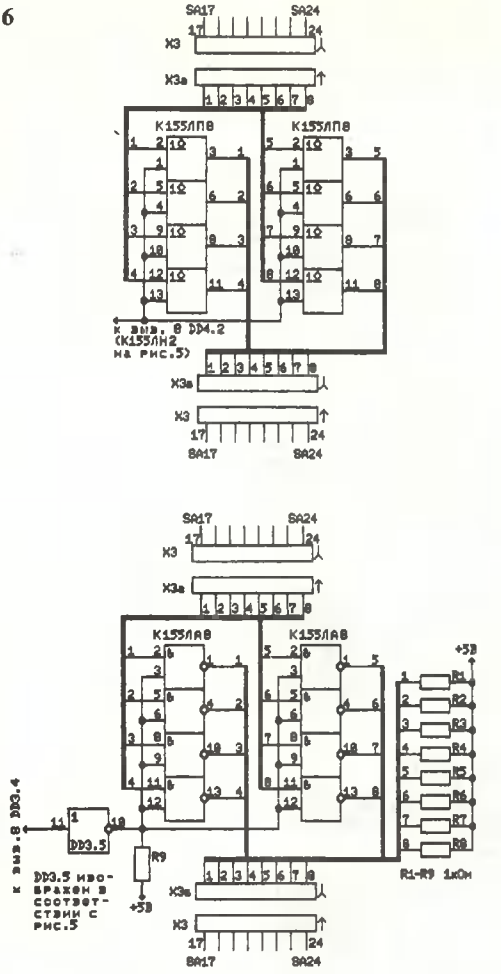
Микросхема DD2 совместно с переключателем SA27 и кнопкой SA28 позволяет использовать непрерывный или пошаговый режимы работы микропроцессора.

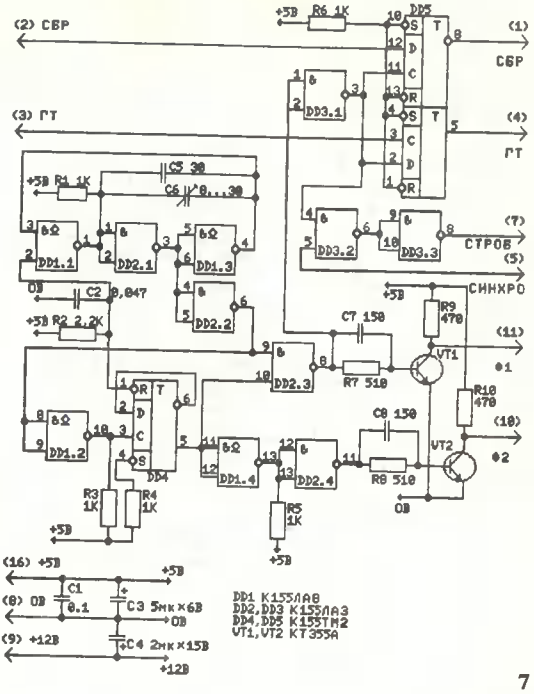
Конструктивно модуль ввода, вывода и управления может быть собран в виде одного блока, подключаемого при помощи штыревого разъема к общей шине; можно выполнить его и в виде четырех отдельных узлов, изображенных на рис. 1. Такой путь наиболее приемлем при кружковой работе, когда каждый кружковец получает задание на изготовление отдельного узла. Таким образом модуль изготавливается во много раз быстрее.

Питание модулей компьютера осуществляется от стабилизированного блока питания со следующими напряжениями: +5 В (сила тока до 1 А), +12 В (0,1 А) и



5





7

—5 В (0,01 А). Узел светодиодной индикации можно питать отдельно нестабилизированным напряжением, в этом случае ток от источника напряжения +5 В уменьшится до 0,2 А.

Микропроцессор должен включаться в следующей последовательности: —5 В, +5 В, +12 В. Выключение производится в обратном порядке. Допускается одновременное включение (и выключение) всех напряжений.

Взаимодействие компьютера с внешними устройствами ввода и вывода (реле, переключатели, электродвигатели, звуковые и световые устройства) осуществляется путем использования свободных адресных линий A12—A15 (на рис. 5 подключение этих устройств не показано).

После изготовления простейшего компьютера, одновременно с изучением принципов его работы и освоением азов программирования, можно постепенно приобретать необходимые детали и собирать дополнительные модули, расширяющие его функциональные возможности.

Так, например, замена тумблерного ввода адреса и данных клавишным, имеющим шестнадцатиричную клавиатуру и семисегментные индикаторы, позволяет с бóльшим удобством вводить программы, написанные на языке ассемблера, а изготовление полной клавиатуры и модулей сопряжения с телевизором позволяет пользоваться Бейсиком. Описание принципа работы компьютера, процедуры его налаживания, рекомендации по дальнейшему совершенствованию путем установки дополнительных модулей (клавиатуры и т. п.), а также особенностей программирования для него будут опубликованы в следующих номерах журнала.

От редакции. Уважаемые читатели, отнеситесь к описанной конструкции со всем возможным уважением! Первый в мире персональный компьютер «Альтаир 8800», поступивший в широкую продажу в 1975 г., был построен на основе аналогичного процессора, имел тумблерный ввод, светодиодный вывод и ОЗУ емкостью 256 байт. Иначе говоря, пользуясь советами только что прочитанной статьи, вы можете собрать нечто лучшее, чем первый ПК, и к тому же значительно дешевле 397 долларов, которые он стоил.

В. ДОБРЯКОВ
Волгоград

Эффектно и эффективно

Повышение интереса к занятиям по информатике зависит от многих факторов, одним из которых является подбор учебного материала. На первых этапах обучения задания для учащихся должны быть небольшие, быстро выполняемые. В таких программных этюдах легко обнаруживается ошибка, учебный материал не «затенен», а эффектное изображение на экране улучшает эмоциональное восприятие изучаемого материала. Из этих этюдов учащиеся составляют «банки» приемов программирования, что способствует творческому осмыслению этих приемов.

Вот конкретные примеры.

Работу операторов цикла можно пояснить на такой программе (для БК-0010, черно-белый монитор):

```
10 CLS
20 FOR A=0 TO 250
30 PRINT "ФРАГМЕНТ УЗОРА"
40 NEXT A
50 END
```

Мои учащиеся назвали ее «САПР-ковры». В строке 30 учащиеся набирают свой фрагмент узора (например, X¤+¤X), и на экране у каждого свой «ковер»; не понравился, придумай другой узор. Не хочешь «ковер», можно «ковровую дорожку».

А вот узоры для «вязания»:

```
10 CLS
20 INPUT "ВВЕДИ ЧИСЛО ЗУБЦОВ";N
30 FOR I=0 TO 2*PI STEP 2*PI/N
40 X=120+80*COS(I)
50 Y=128+80*SIN(I)
60 X1=120+80*COS(I+PI*4/5)
70 Y1=120+80*SIN(I+PI*4/5)
80 LINE-(X,Y)
90 LINE-(X1,Y1)
100 NEXT I
110 END
```

Здесь достаточно и математики, и материала по Бейсику. Этюд дает широкие возможности поэкспериментировать, он может найти применение в других программах, например может быть использован в качестве заставки.

Изучение функции RND можно провести на «взрыве»:

```
10 CLS
20 FOR K=1 TO 100
30 X=INT(RND(1)*250)
40 Y=INT(RND(1)*240)
50 CIRCLE (X,Y),2
60 NEXT K
70 END
```

Понимание можно проверить такими заданиями: сместить центр «взрыва», изменить длину лучей, изменить количество лучей, написать аналогичную программу и т. д.

Последнее задание можно выполнить, например, так:

```
10 CLS
20 FOR K=1 TO 50
30 X=INT(RND(1)*249)+1
40 Y=INT(RND(1)*239)+1
50 LINE (125,120)-(X,Y)
60 NEXT K
70 END
```

Изменился оператор 50, и на экране вместо «взрыва» — «кипение» или «газировка» (придумать название — это тоже творчество).

Одно из применений математических функций можно показать на таком этюде:

```
10 CLS
20 INPUT N
30 FOR K=0 TO 2*PI STEP 2*PI/N
40 X=120+40*COS(K)
50 Y=180+40*SIN(K)
60 LINE-(X,Y)
70 NEXT K
80 FOR K=0 TO 2*PI STEP 2*PI/N
90 X=120+40*COS(K)
100 Y=180+40*SIN(K)
110 LINE (X,Y)-(120,10)
120 NEXT K
130 END
```

На экране строится пирамида. Это интереснее, чем вычисление значений функций, хотя и от вычислений отказываться нельзя. Учащимся можно предложить такие задания: объясните работу программы; какой комментарий можно поместить в оператор 20; составьте блок-схему; каково назначение первого (второго) цикла; напишите программу построения многоугольников и т. п.

Один из вариантов программы построения многоугольников может выглядеть так:

```
10 CLS
20 INPUT N
30 FOR K=0 TO 2*PI STEP 2*PI/N
40 X=120+80*COS(K)
50 Y=120+80*SIN(K)
60 LINE-(X,Y)
70 NEXT K
80 END
```

Эта программа позволяет продемонстрировать переход многоугольника в окружность.

Следующие два этюда позволяют показать приемы имитации движения на экране.

```
10 REM МАЯТНИК
20 FOR I=30 TO 240 STEP 20
30 LINE (120,10)-(I,150)
40 CIRCLE (I,150),5
50 CLS
60 NEXT I
70 FOR I=240 TO 30 STEP -20
80 LINE (120,10)-(I,150)
90 CIRCLE (I,150),5
100 CLS
110 NEXT I
120 GOTO 20
```

Выход из программы клавишей **СТОП**.

Идея следующей программы заимствована у В. А. Каймина («Звездное небо»). Наша программа получила название «Хвойный лес». Мы ее даем для самостоятельной разработки. Такого типа задания очень удобны при обучении составлению сценариев программ.

```
10 CLS
20 FOR I=1 TO 200
30 PSET (I,I)   'нарисовать диагональ
40 NEXT I
50 FOR K=200 TO 0 STEP -1
60 PRESET (K,K)   'стереть линию
70 NEXT K
80 GOTO 20
Выход из программы клавишей СТОП
```

```
10 CLS
20 INPUT "ЧИСЛО ЕЛОЧЕК В ЛЕСУ";K
30 CLS
40 FOR I=1 TO K/2
50 X=INT(RND(1)*250)
60 Y=INT(RND(1)*240)
70 PSET (X,Y),0
80 DRAW"BM+6,1 ND8 NF6 NG7 U3 NF4 NG4
   U3 NF3 NG4 U3 NF2 NG2 U3 NF1 NG1 U3"
90 DRAW"BM+6,1 ND8 NF6 NG7 U3 NF4 NG4
   U3 NF3 NG4 U3 NF2 NG2 U3 NF1 NG1 U3"
100 NEXT I
102 FOR I=1 TO 500
104 K=INT(RND(1)*255)
106 M=INT(RND(1)*240)
108 PSET (K,M),1
110 NEXT I
120 ? AT(7,23)"ДРУГОЙ ЛЕС Д/Н"
130 INPUT N$
140 IF N$="D" THEN 10
150 END
```

Интересные «изюминки» приемов программирования можно найти в методических пособиях, подготовленных М. Е. Степановым (НИИ школ Министерства народного образования РСФСР).

П. ПАНКОВ
г. Фрунзе

Глазомерные задачи

Преподавание математики имеет целью не только усвоение определенного объема знаний, но и развитие навыков, которые называют «математической интуицией», пространственным воображением. Опишем разрабатываемую в РИУУ МНО Киргизской ССР и КГУ методику, имеющую целью связать практический опыт и математические знания.

Глазомерными назовем такие задачи, где ответ может быть только приближенным, нельзя пользоваться какими-либо предметами (бумагой, карандашом, линейкой, микрокалькулятором, справочником) и на решение отводится небольшое время (15—30 с). При этом человек может представить себе математический образ (или оценить чертеж, изображение на дисплее), но не успевает сделать в уме достаточно точные вычисления.

Через X_0 будем обозначать ответ, вычисляемый ЭВМ с абсолютной или достаточно большой точностью, через X — введенный человеком. Предлагались следующие варианты требований: указать X, наиболее близкий, по мнению человека, к X_0; указать такой наиболее узкий интервал X, что, по мнению человека, $X_0 \in X$.

В первом случае критерий оценки по отклонению $D=|X-X_0|$ или по относительному отклонению $P=|X/X_0-1|$: $P<0{,}1$ — отлично, $0.1 \leqslant P<0.2$ — хорошо, $0.2 \leqslant P<0.4$ — удовлетворительно. Во втором случае если введенный интервал не содержит решения ($X_0 \notin X$), то начисляются штрафные баллы (сумма баллов уменьшается), иначе ($X_0 \in X$) сумма баллов увеличивается на величину, обратно пропорциональную ширине интервала X.

И условие и ответ могут отображаться как в графическом, так и в символьном виде.

Приведем несколько задач, реализованных на ПЭВМ «Агат», «Ямаха» и «Искра-1030».

1. На дисплее высвечиваются (случайное) кольцо и маленький квадрат. Требуется увеличить квадрат настолько, чтобы его площадь

X была равна площади кольца X_0.

2. Указать приближенно (в числовом виде) корень уравнения $X^3+B\cdot X=C$, где B, C — случайные натуральные числа.

3. На дисплее высвечивается (случайный) треугольник. Требуется движением метки по лучу указать отрезок, длина X которого равна периметру X_0 треугольника.

4. На дисплее высвечиваются три (случайных) точки (случайной) параболы с вертикальной осью. Требуется движением метки по экрану указать ее вершину.

5. Указать (в градусах) приближенно угол в треугольнике, если даны прилежащие стороны A, B и противолежащая C — натуральные (случайные) числа.

6. На дисплее высвечиваются (случайный) треугольник и вертикальная прямая рядом с ним. Требуется горизонтальным сдвигом этой прямой разделить площадь треугольника пополам.

7. Даны (графически) вектора A_1, A_2 и B. Требуется указать (в числовом виде) коэффициенты разложения по базису $B=X_1\times A_1+X_2\cdot A_2$. (Ответ, введенный человеком, представляется в графическом виде.)

8. По данному (случайному) набору точек графически указать центр тяжести набора.

Приведенному выше неформальному определению также удовлетворяет задача «Стрелы» из книги Г. Клеймана «Школы будущего: компьютеры в процессе обучения».

9. На отрезке [0, 1] на экране дисплея располагается (случайный) кружок. Нужно ввести обыкновенную дробь, показывающую положение кружка (с точностью до его радиуса).

Данную методику можно применить и к другим учебным предметам — физике, географии, биологии.

О построенных таким путем задачах просьба сообщать по адресу: 720888, г. Фрунзе, ул. Краснооктябрьская, 257, РИУУ МНО Киргизской ССР, кафедра управления и информатики, Панкову П. С.

А. КОЧЕРГИН

Задача о слухах

Все больше вестей о событиях в стране и в мире приносят нам средства массовой информации, однако продолжают еще ходить слухи. При всей доморощенности этого способа распространения новостей он может быть весьма эффективным и ненамного уступать даже телевидению. Попробуем смоделировать процесс распространения слухов, чтобы внимательнее рассмотреть, как он происходит.

Итак, сколько человек узнают новость через трое суток, если первоначально ее знал лишь один?

Для начала — простейший вариант: будем исходить из того, что слухи передаются 10 ч в сутки — в среднем одна «передача» в два часа; первый обладатель слуха передает его двум знакомым, каждый из них — двум другим и т. д.

```
10 A=1 : N=1
20 FOR T=2 TO 30 STEP 2
30 A=2*A
40 N=N+A
50 PRINT "T="T,"N="N
60 NEXT T
70 END
```

Здесь в строке 20 задан период моделирования (30 ч, т. е. трое суток), в строке 30 подсчитывается число людей, получивших информацию на данном этапе (на следующем они будут ее распространять), в строке 40 — общее число «информированных».

Теперь вспомним, что люди бывают разные, в том числе и по параметру общительности. Допустим, что «болтуны» (их B%) передают новость четверым, а «молчуны» (их M%) не передают ее никому.

```
10 INPUT "КАКОВ ПРОЦЕНТ 'МОЛЧУНОВ'";M
20 INPUT "КАКОВ ПРОЦЕНТ 'БОЛТУНОВ'";B
30 A=1: N=1
40 FOR T=2 TO 30 STEP 2
50 AB=A*B/100
60 AN=A*(100-B-M)/100
70 A=AB*4+AN*2
80 N=N+A
90 PRINT "T="T,"N="INT(N+0.5)
100 NEXT T
110 END
```

Здесь в строке 50 находится число «болтунов» среди новоиспеченных обладателей слуха, в строке 60 — число «нормальных» передатчиков. При выводе N округ-

ляется до целого (люди не винтики!).

До сих пор каждый узнавший слух передавал его (в меру своей общительности) дальше. В жизни же интерес вызывает не любая информация а только новая. Поэтому из числа передатчиков будем исключать тех, для кого новость не нова, иначе говоря узнавших ее вторично. Число таких людей зависит от скорости распространения слуха и населения города. Предыдущая программа изменится лишь на несколько строк.

```
5 INPUT "КАКОВА ЧИСЛЕННОСТЬ
                          НАСЕЛЕНИЯ";S
...
25 K=0
...
73 IF T>4 THEN K=1.6*N/S: X=K*A: A=A-X
76 IF K>=1 THEN 110
...
90 PRINT "T="T,"K="INT(K*100+0.5);
                        "N="INT(N+0.5)
```

Здесь K — коэффициент, учитывающий число людей, получивших информацию повторно; 1.6 — эмпирически подобранный поправочный коэффициент. Поправка в строке 73 начинает вводиться с T=6, так как раньше слух не может вернуться к уже знающим его.

В начале процесса вероятность «возвратного» попадания слуха очень мала, но постепенно возрастает, из-за чего скорость распространения резко уменьшается. В жизни этот процесс, кстати, усиливается параллельной «дезактивацией» «болтунов» и их переключением на новый слух (свято место пусто не бывает!), в итоге часть населения так и остается неинформированной. Анализ размеров этой части в зависимости от различных условий представляет самостоятельный интерес.

Следующим шагом усложнения задачи может быть организация вывода результатов в привлекательной графической форме.

Точность разобранных моделей, само собой разумеется, далека от совершенства; можно учитывать еще погоду, наличие «конкурирующих» новостей, эпидемию гриппа, показ «Рабыни Изауры» и т. п. С другой стороны, с помощью моделей такого типа можно изучать распространение эпидемий, ход реакции в ядерных реакторах и бомбах, рост колоний микроорганизмов и т. д.

Х. ЯДГАРОВ

Программа — тренер

На первых практических занятиях у наших (думаю, и у других) учеников нелегко формировалась привычка к восприятию масштабов при работе в различных режимах графического экрана. Чтобы преодолеть это, было разработано несколько очень простых игр, после 10—15 мин «работы» с которыми очень хорошо запоминались масштабы различных режимов графического экрана (на КУВТ «Ямаха» их шесть).

Суть приводимой в качестве примера игры такова: на экране появляется круг со случайным радиусом. Ученик должен определить визуально радиус круга, а ЭВМ оценивает, насколько точно это сделано.

Обычно к концу первого урока ученик осваивает масштабы данного режима и начинает получать максимальные оценки.

```
10 REM ИГРА "УГАДАЙ РАДИУС"
20 D1=1
30 FOR I=1 TO 5
40 R=FIX(RND(1)*60):NEXT
50 X=128
60 Y=96
70 CLS
80 INPUT "В КАКОМ РЕЖИМЕ БУДЕТЕ
                          РАБОТАТЬ";N1
90 IF N1>8 OR N1<2 THEN 80
100 R=FIX(RND(1)*60)
110 TIME=0
120 SCREEN N1
130 CIRCLE(X,Y),(R),8
140 IF TIME<130 THEN 130
150 SCREEN 0
160 INPUT "РАДИУС КРУГА";K:GOSUB 330
170 PLAY"v15t255l64n66n70"
180 IF K=R THEN PRINT "ХОРОШО
   ИГРАЕТЕ. НАБРАЛИ 10  ОЧКОВ":M1=M1+10
190 PLAY"v1116gfedc"
200 T=ABS(K-R)
210 IF T>10 THEN PRINT "ОШИБЛИСЬ"
220 IF T<10 THEN M=10-T:PRINT
          "НЕМНОГО ОШИБЛИСЬ. НАБРАЛИ";
                    M;"ОЧКОВ":M1=M1+M
230 PRINT "БУДЕТЕ ИГРАТЬ ЕЩЕ? (Д/Н)"
240 A$=INKEY$
250 IF A$="D" OR A$="d" OR A$="д" OR
          A$="Д" THEN D1=D1+1:GOTO 100
260 TIME=0
270 IF A$="n" OR A$="N" OR A$="н" OR
     A$="Н" THEN CLS: PRINT "ИЗ";D1*10;
```

```
      "ВОЗМОЖНЫХ ОЧКОВ ВЫ ПОЛУЧИЛИ";M1;
                          "ОЧКОВ" ELSE 240
280 IF TIME<130 THEN 280
290 CLS
300 PLAY"t120v11o4l16cegc5cco4gc5cc",
              "t120v12o4l16egc5ceecee",
               "t120v11o3l16go4ceggegg"
310 PRINT "ДО СВИДАНИЯ"
320 END
330 REM
340 TIME=0: SCREEN N1
350 CIRCLE (X,Y),R,8
360 CIRCLE (X,Y),K,10
370 IF TIME<100 THEN 370
380 SCREEN 0: RETURN
```

От редакции. А вот аналогичная программа для БК-0010; попробуйте, уважаемые читатели, улучшить ее, сократить и т. п.

```
10 CLS
20 ?CHR$(&H94)CHR$(&H9E)CHR$(&H91)
30 POKE 112%,15360%+PEEK(132%)
40 ? " ** УГАДАЙ РАДИУС ОКРУЖНОСТИ **
50 CLS
60 D1%=1%
70 M%=0%
80 X%=128%
90 Y%=128%
100 CLS
110 M%=M%+M1%
120 FOR I%=0%TO7%
130 R%=FIX(RND(1%)*60%)
150 NEXT I%
160 LOCATE 2%,3%
170 CIRCLE (X%,Y%),R%,,,,1.4
180 INPUT "Ответьте, каков радиус";K%
190 GOSUB 460
200 IF K%=R%TH210EL300
210 M1%=10%
220 ? "Хорошо! Наврали целых 10 очков!
230 FOR B%=0%TO500% 'звуковой
240 POKE -50%,224%  'сигнал
250 FOR U%=0%TO5%   '
260 NEXT U%         '
270 POKE -50%,160%  '
280 NEXT B%         '
290 GOTO 350
300 T%=ABS(K%-R%)
310 IF T%>10%TH?AT(2%,4%)"Очень сильно
                          ошиблись!"EL330
312 M1%=0%
320 GOTO 350
330 IF T%<10%TH M1%=10%-T%
340 IF T%,10%TH?AT(2%,4%)"Чуть ошиблись
                    и наврали";M1%;"очков"
350 ? AT(2%,5%)"Будете играть еще?
                                    (д/н)"
360 A$=INKEY$
370 IF A$=""TH360
380 IF A$="D"ORA$="Д"ORA$="d"ORA$="д"
                                  TH390EL410
390 D1%=D1%+1%
400 GOTO 100
410 CLS
420 M%=M%+M1%
430 ? AT(2%,10%)"Из";D1%*10%;"возможных
    очков"AT(2%,11%)"вы получили";M%;",
          "AT(2%,12%)"что составляет";
          FIX((M%*10%)/D1%;"% успеха!"
440 ? AT(5%,21%)"До свидания!.."
450 END
460 CIRCLE (X%,Y%),K%,,,,1.4
470 PAINT STEP(INT((K%/1.4)-1%),0)
480 RETURN
```


В. БОНДАРОВСКАЯ, Н. ПОВЯКЕЛЬ
г. Киев, НПО «Горсистемотехника»

Психолого-эргономическое обеспечение программных средств

Компьютеризация образования предполагает создание новой методологии и технологии обучения, широкое применение в образовании принципов психологии и управления. Опубликованная недавно психологическая концепция использования компьютерных учебных средств в процессе обучения предусматривает изменение содержания и методов обучения, проектирование компьютеризованной деятельности учащихся [1].

Новые возможности вызывает развитие новых свойств программного обеспечения, особенно форм общения человека с ЭВМ. Для педагогических программных средств (ППС) эта проблема стоит особенно остро. Необходимо обеспечить психологическую естественность деятельности учащегося с ЭВМ, адекватность программы целям и функциям обучения, удобство работы учащегося с ЭВМ и сохранение его здоровья. Психологичность и эргономичность являются одними из важнейших характеристик качества ППС. Широко пропагандируемая в настоящее время «дружественность программного обеспечения» [2] как раз и предполагает наличие психолого-эргономической поддержки разработки программных средств.

Применение ППС расставляет особые акценты между психологической и эргономической поддержкой дидактических целей. Психологическая естественность в соответствии с возрастными возможностями учащегося теснейшим образом связана с обеспечением таких эргономических требований,

как воспринимаемость информации, создание необходимого темпа диалога, выделение особых зон для подсказки и т. п. Как отмечает Г. С. Цейтин [3], любая разработка программного обеспечения включает в себя задачу проектирования деятельности будущего пользователя создаваемой системы.

В практике автоматизации вопросы проектирования деятельности будущего пользователя обычно решаются стихийно, в лучшем случае на основе опыта авторов системотехнического обеспечения, а чаще всего исходя из случайных соображений. Более того, проект деятельности пользователя не входит в состав документации на автоматизированную систему, не является законченным продуктом ее разработки. И как следствие отсутствуют психологически и эргономически обоснованные решения по таким важным вопросам, как определение класса решаемых пользователем задач, проектирование языка его взаимодействия с ЭВМ, выбор вида диалога, разработка дисплейных форматов, что приводит, как правило, к низкой мотивации у пользователей при решении задач с применением ЭВМ, к снижению эффективности их деятельности, повышенной утомляемости, к возникновению трудностей в освоении средств вычислительной техники.

В [4, 5, 6] предложен проектный программно-исследовательский подход к созданию психолого-эргономического обеспечения технических и программных средств деятельности (в данном случае учебной) пользователя. Как показал наш опыт создания эргономического обеспечения практически реализованных видеотерминальных систем, только на пути совместного участия специалистов различных предметных областей (педагогов, психологов, эргономистов, разработчиков программных средств, представителей пользователя) может быть адекватно решен вопрос о разработке проекта деятельности пользователя (учащегося и преподавателя).

Начальный этап разработки проекта деятельности учащегося — пользователя ЭВМ включает следующие проектные системотехнические, психологические и эргономические моменты:

системно-психологическую характеристику учащегося-пользователя;

логико-психологическое описание класса решаемых с помощью ППС задач;

перечень программных поддержек основных стандартных процедур решения указанных задач;

описание структуры компьютеризированной деятельности учащегося, включающее цели, действия, процедуры, средства реализации, эффективные стратегии осуществления информационных технологий.

На последующих этапах проект деятельности развивается, уточняется и модифицируется в соответствии с ограничениями, задаваемыми социальными и организационно-экономическими критериями.

При разработке ППС и реализации проекта деятельности учащегося и преподавателя, выступающих в качестве пользователей ЭВМ, следует учитывать ряд психологических принципов, а также эргономических требований.

В качестве таких психологических принципов следует рекомендовать, например, следующие.

Следует учитывать индивидуальные различия учащихся, в частности предусматривать возможность обучения в различном темпе и получения помощи различной степени подробности.

Необходимо иметь в виду, что важную роль играет форма представления знаний о пространстве задачи, решаемой учащимся. При выборе форм представления знаний на экране компьютера необходимо исходить не только из содержания учебной деятельности, но и из тех возможностей, которые предоставляет компьютер для реализации эффективных стратегий решения и достижения таких дидактических целей, которые при «ручной» технологии обучения оказываются недостижимыми.

В связи с особым ритмом общения человека с ЭВМ особую роль приобретает проблема понимания текстов. Это касается не только понимания текстов программ как учебного материала на уроках информатики, но и понимания тех учебных текстов, которые предъявляются учащемуся на экране компьютера.

Необходимо исследовать, как приобретаются новые навыки и умения при использовании такого нового средства обучения, как компьютер.

При разработке ППС эргономические требования могут быть представлены к:

процедуре взаимодействия пользователя с ЭВМ;

видам диалога пользователя с ЭВМ;

проектированию дисплейных форматов;

контролю ошибок учащегося;

временным параметрам диалога учащегося с ЭВМ;

организации информации на экране;

кодированию информации на экране;

языкам взаимодействия пользователя с ЭВМ.

Требования к процедуре взаимодействия пользователя с ЭВМ. Процедура взаимодействия учащегося с ЭВМ должна основываться на характеристиках структуры учебной деятельности, которая соответствует целям

обучения и определяется в значительной степени средствами ее реализации.

Общие требования к характеристикам структуры деятельности учащегося включают:

определение содержания обучения;

выделение выполняемых учащимся заданий;

определение общих и конкретных целей деятельности учащегося при выполнении каждого вида заданий;

выделение действий и операций, которые должен выполнять учащийся в режиме взаимодействия с ЭВМ;

определение характеристик технических и программных средств его деятельности.

При разработке ППС следует четко определить уровень подготовленности учащихся — пользователей ЭВМ.

С целью использования имеющихся эргономических требований целесообразно соотнести характеристики контингента предполагаемых учащихся с характеристиками того типа пользователей, которые пользуются аналогичными процедурами диалога.

Необходимо определить те знания и умения, которые потребуются учащемуся при эксплуатации ЭВМ, определить язык взаимодействия, навыки работы с функциональной клавиатурой, навыки работы с машинной графикой и т. п. Поскольку учащийся часто оказывается так называемым казуальным пользователем, должны учитываться требования к повышению ясности и понятности ответа от ЭВМ, ко времени ответа от ЭВМ на запрос учащегося.

Особую роль играет определение характеристик заданий учащегося. Их следует выделять с учетом того, что компьютер более пригоден для хранения больших массивов информации, быстрого и точного ее воспроизведения, быстрой и точной переработки информации, позволяет осуществить визуализацию разнообразной информации. Следует выделить задания и функции, которые совместно выполняют учащийся и ЭВМ. Если учащийся имеет дело с трудно интерпретируемыми дисплейными форматами, слишком сложны его функции в системе. Если же задания спроектированы так, что на учащегося возлагаются функции, ведущие к монотонии его учебной деятельности, то появляется большое количество ошибок, повышается его утомляемость.

Требования к выбору вида диалога. В зависимости от типов заданий, характеристик структуры деятельности учащегося и особенно технических и программных средств необходимо выбирать виды диалога учащегося с ЭВМ.

Конкретизация вида диалога должна производиться в ходе разработки процедуры диалога как части эргономического обеспечения ППС на основе проекта деятельности учащегося — пользователя ЭВМ. Определение приемлемого вида диалога [4, 7, 8] должно производиться при эргономическом проектировании с участием представителей пользователей.

Следует учитывать, что для учащегося, как для наименее обученного пользователя, рекомендуется применение вопросно-ответного вида диалога, базирующегося на естественном языке с элементами диалога, основанного на выборе из меню.

Следующий шаг — проектирование сценария диалога. При этом следует предусматривать такую последовательность ввода сообщений, которая была бы наиболее близка к входному документу.

Не рекомендуется требовать от учащегося выдачи подтверждающих сообщений слишком часто.

Не следует подчинять ритм работы учащегося ритму работы ЭВМ.

Необходимо информировать учащегося в ходе диалога о действиях, которые потребуются от него на следующем шаге.

Необходимо предусматривать в процедуре диалога возможность приостановки работы в любой момент, а также возможность возобновить работу, если это понадобится учащемуся с нужного места.

Необходимо предусматривать возможность напоминания учащемуся о действиях, имевших место на предыдущих стадиях диалога.

Целесообразно предусматривать различные уровни общения с ЭВМ для учащихся с разной подготовкой, от подробного, снабженного большим количеством подсказок, до сжатого, содержащего лишь основные необходимые шаги диалога.

Сообщения, предъявляемые учащемуся на экране дисплея, должны быть логически последовательны, отнесены к структуре его деятельности, последовательно решать подзадачи, предусмотренные содержанием обучения.

Близкие по смыслу единицы информации должны быть сгруппированы так, чтобы повысить читабельность текстов.

Информация должна быть структурирована вплоть до разделения на последовательность предъявляемых учащемуся кадров. В случае наличия большого количества необходимой для предъявления информации рекомендуется разделить ее на несколько кадров, а не предъявлять всю на одном экране.

На экране должна предъявляться та информация, которая имеет отношение к тем действиям учащегося, которые он выполняет

в данный момент. При этом следует учитывать, что рекомендуется избегать чтения текстов с экрана. В случае, когда требуется изучение значительного текстового материала, целесообразно использовать учебные пособия. Процедуру взаимодействия учащегося с ЭВМ целесообразно использовать только тогда, когда возможности компьютера позволяют решить новые дидактические задачи.

Требования к проектированию дисплейных форматов. При разработке дисплейных форматов следует учитывать следующие требования:

не следует заполнять весь экран информацией или одной фигурой;

важнейшую информацию желательно располагать в середине экрана;

повторяющуюся информацию следует размещать на тех же самых позициях;

ориентация текстовой информации на экране дисплея должна быть вертикальной;

расстояние между строками должно быть больше, чем высота символов (текстовую информацию лучше представлять через строку);

при проектировании длины строк следует учитывать возраст испытуемых (например, для пользователей — дошкольников и младших школьников желательны более короткие строки).

Требования к организации контроля ошибок учащегося. Организация контроля ошибок учащегося при работе с ПЭВМ должна базироваться на психологических принципах и проекте его деятельности, а также на устранении тех психологических факторов, которые определяют появление ошибок учащегося, среди которых:

перегрузка «пользовательского канала» (учащемуся предъявляется одновременно слишком много сообщений);

пользовательская скука (задания должны быть не слишком сложны, но и не слишком просты, особенно для подготовленного для работы с ПЭВМ учащегося);

недостаток мотивации (учащийся нуждается в наличии интереса к работе);

неадекватные инструкции для управления непредвиденными ситуациями (когда ошибки имеют место, учащиеся должны быть немедленно о них информированы).

При проектировании процедуры обнаружения и диагностики ошибок рекомендуется избегать зашифрованных сообщений об ошибках.

Следует избегать в процессе исправления ошибок повторного ввода правильно введенных ранее данных.

Целесообразно предусматривать вариантность в обеспечении учащихся нужной информацией при обработке ошибок:

более опытным учащимся в работе с ПЭВМ необходимо лишь указать на наличие ошибки, например, блокировкой клавиатуры, изменением яркости или специальным символом;

неподготовленному учащемуся необходимо подробно объяснить характер ошибки и пути ее исправления.

Сообщения об ошибках должны быть несложными, четкими и максимально понятными, а также точными и вызывающими доверие.

Требования к временным параметрам диалога учащегося с ЭВМ. Большую роль в обеспечении эффективного использования ЭВМ играет психологически и эргономически обоснованный выбор времени ответа ЭВМ на различные запросы пользователя [4, 7, 10].

При определении временных параметров процедуры диалога необходимо стремиться к постоянству времени реакции системы на однотипные запросы учащегося и минимизации времени его ответа.

Время реакции ЭВМ на различные запросы учащегося должно проектироваться с учетом рекомендаций психологов относительно подготовленности, возрастных особенностей учащихся, специфики различных запросов.

Вопросы проектирования временных параметров диалога должны решаться в ходе разработки эргономического обеспечения каждой конкретной учебной программы с учетом психологических требований к организации и содержанию компьютеризированного обучения.

Требования к организации информации на экране. Можно выделить целый ряд эргономических требований к организации информации на экране:

информация, предъявляемая на экране, должна быть понятной, логически связной, распределенной на группы по содержанию и функциональному назначению;

при организации информации на экране следует избегать избыточного кодирования и неоправданных, плохо идентифицируемых сокращений;

рекомендуется минимизировать на экране использование терминов, относящихся к ЭВМ, вместо терминов, привычных для учащихся;

не следует для представления информации использовать краевые зоны экрана;

на экране должна находиться только та информация, которая обрабатывается учащимся в данный момент;

при редактировании текста рекомендуется выводить информацию на экран таким образом, чтобы оба абзаца текста, актуальных в данный момент, были одновременно представлены на экране.

В современных программных средствах ис-

пользуется ряд приемов для выделения части информации на экране: переструктурирование информации и выделение зон, окон для выделяемой части информации, а также инверсное изображение для части информации и различные эффекты, привлекающие внимание пользователей (мелькание и др.). Использование этих приемов должно быть психологически обосновано особенно для ППС, функционально обусловлено и эргономично. Рекомендуется:

вопросно-ответные сообщения и подсказки помещать в верхней части экрана, выделяя явным образом отведенную для этого зону, например отделяя ее горизонтальной линией от основной информации на экране;

различные виды сообщения необходимо отделять друг от друга в зоне вспомогательной информации. Например, можно рекомендовать применять инверсное изображение для подсказок;

зоны размещения на экране вспомогательной информации должны быть четко идентифицируемы — зона подсказок, зона комментариев, зона управляющих сообщений, зона для сообщений об ошибках;

при зонировании экрана допускается изменение масштаба знаков в отдельной зоне;

эффекты, привлекающие внимание учащегося — пользователя ПЭВМ (мелькание, повышенная яркость, обратный контраст), следует применять строго в соответствии с проектом деятельности пользователя, только в тех случаях, когда это необходимо и психологически обосновано.

Требования к кодированию информации на экране. Большое внимание в настоящее время уделяется цветовому кодированию изображения. Оно широко используется при разработке ППС.

Кодирование цветом позволяет структурировать информацию, обеспечить большую ее читабельность и идентифицируемость, а также выделять важную информацию. Кодирование цветом обеспечивает малое время обнаружения и идентификации информации на экране. Цветовым кодированием следует пользоваться при необходимости быстрого обнаружения местонахождения нужного символа на экране с неупорядоченной информацией или на плотно загруженном экране; для выделения сигнала тревоги, предупреждающей информации об изменении какого-либо параметра; для выделения подсказки; при группировке однородной или разнородной информации; при организации многооконного интерфейса.

Выбор цветов и их сочетаний определяется возможностями конкретной ЭВМ и оказывает воздействие на интеллектуальную деятельность пользователя, на его работоспособность и эффективность работы.

При определении сочетаемости цветов для фона и символов необходимо иметь в виду: ограниченность человеческих возможностей в размещении цветов; использование дополнительных цветов часто вызывает зрительную утомляемость и раздражение пользователя; значения цветового контраста должны находиться в пределах значений допустимых для яркостного контраста; многие пользователи имеют красно-зеленую или желто-синюю аномалию зрения; монохромные дисплеи по-разному отражают разные цветовые контрасты, используемые в цветных дисплеях.

Большие возможности предоставляет в настоящее время многооконный способ представления информации на экране.

Многооконный вывод предполагает наличие нескольких зон предъявления информации, отделенных друг от друга: в пространстве, цветом, формой и/или другими способами представления информации.

При организации окон на экране дисплея не рекомендуется использовать краевые зоны экрана, а также применять в большом количестве эффекты, привлекающие внимание пользователя (мелькания, повышенная яркость, обратный контраст и др.). Такие эффекты следует применять только в психологически оправданных ситуациях.

При многооконном выводе рекомендуются четко идентифицированные зоны размещения вспомогательной информации на экране дисплея: зона подсказок, зона комментариев, зона управляющих сообщений, зона для сообщений об ошибках. Вопросно-ответные сообщения и подсказки рекомендуется помещать в верхней части экрана, выделяя явным образом отведенную для этого зону, например отделяя ее горизонтальной линией от основной информации на экране.

При использовании способа цветового кодирования изображения в организации многооконного вывода следует руководствоваться рядом рекомендаций:

необходимо минимизировать количество цветов, используемых на экране одновременно;

цвет переднего плана и цвет фона должны находиться в сочетании между собой и с цветом тех символов, которые должны быть выделены на экране. Исходя из этого не рекомендуется использовать яркие цвета для границ окон и для заглавий;

окна следует разделять между собой цветом фона;

при кодировании окон следует учитывать три функции цвета: выделение, задний план и нормальное чтение;

выделяемые объекты, например сообще-

ния об ошибках, рекомендуется окрашивать в цвета, используемые для предостережения.

При текстообработке для кодирования информации, предъявляемой пользователю непрерывно, но необходимой ему только изредка, должны применяться нейтральные цвета.

Требования к разработке языковых средств взаимодействия. Помимо перечисленных психологических и эргономических требований к характеристикам педагогических программных средств важное значение имеет и разработка психологического обеспечения языково-программных средств взаимодействия учащегося-пользователя с ЭВМ: языков мнемоник и пиктограмм на экране дисплея и клавиатуре; языков программирования, языков диалога, языков отладки, языков запроса и др.

Для «психологизации» языковых средств взаимодействия с ЭВМ необходимо привлечение психолога к разработке языков на всех этапах проектирования и реализации.

Основополагающим в выработке критериев и принципов их проектирования является проект деятельности пользователя. При проектировании, выборе и оценке языков взаимодействия учащихся с ПЭВМ рекомендуется рассматривать три группы критериев: критерии, отражающие уровень проблемной ориентированности языка; критерии, отражающие уровень психологической ориентированности языковых средств; критерии, отражающие уровень процессуальной или процедурной ориентированности языковых средств [4, 10, 11].

В заключение следует отметить, что наиболее важное условие эффективного использования ППС без явлений повышенного утомления учащихся — это включение психологических и эргономических средств, начиная с самых ранних этапов их создания, вплоть до кардинального пересмотра подхода к применению ЭВМ в образовании и содержания и структуры деятельности учащегося и преподавателя.

Литература

1. Логико-психологические основы использования компьютерных учебных средств в процессе обучения / В. В. Рубцов, В. Н. Каптелинин, В. А. Львовский и др. // Информатика и образование. 1988. № 3. С. 3—16.
2. *Иванов Г. Н.* Дружелюбный компьютер / Сер. «Вычислительная техника и ее применение». М.: Изд-во «Знание», 1989. № 3. С. 3—21.
3. *Цейтин Г. С.* Психологические задачи в разработке программного обеспечения // Микропроцессорные средства и системы. 1984. № 3. С. 48—59.
4. *Бондаровская В. М., Повякель Н. И.* Психология и программное обеспечение ЭВМ: теория и практика. Киев: Знание, 1988. 16 с.
5. *Бондаровская В. М., Повякель Н. И., Мирончено С. А.* Человек и видеотерминал: проектный программно-исследовательский подход // Эргономика рабочего места / Тезисы докладов ученых и специалистов стран — членов СЭВ на VI Международной конференции по эргономике. М.: ВНИИТЭ. С. 8—9.
6. *Бондаровская В. М., Мирончено С. А., Повякель Н. И., Чачко С. А.* Повышение потребительских свойств технических средств СМ ЭВМ в процессе их разработки // Приборы и системы управления. 1985. № 4. С. 18—20.
7. *Stewart T.* The Software Interface // Ergonomics in Office Automation.: Ericson. Stockholm. 1988. P. 77—99.
8. *Martin J.* Design of Man — Computer Dialogues. New. York. 1973.
9. *Guynes J. L.* Impact of System Response Time on State Anxiety // Communication of the ASM. March 1988. Volume 31. N 3. P. 342—347.
10. *Повякель Н. И.* Влияние языков программирования на целеобразование в деятельности пользователя ЭВМ: Автореф. дис. ... канд. психол. н. М., 1988. 25 с.
11. *Повякель Н. И.* Целеобразование в психологическом обеспечении программных средств пользователя ЭВМ // Психологические проблемы создания и использования ЭВМ. М.: Изд-во МГУ, 1985. С. 79—81.

Научно-технический кооператив «КОМП»

Новые высокоскоростные модемы для IBM-совместимых машин заполнят пробел на отечественном рынке

Разработаны два варианта встраиваемого модема: «С-9600 М1», «С-9600 М2» для IBM PC/XT/AT и для ДВК-2/3/4 соответственно;

оба стандартных модема поддерживают одинаковый протокол обмена, позволяющий подстраиваться под состояние телефонной линии;

модемы выполнены в виде полуплат и вставляются в свободные разъемы расширений компьютеров;

прямое включение в телефонную линию;

реализован автовызов и автоответ;

максимальная скорость работы 9600 бит/с в условиях отличного состояния телефонной линии, на местной или выделенной линии;

средняя скорость (в Москве, а также по междугородным линиям) составляет 2400 бит/с;

протокол обмена и коррекции ошибок обеспечивает 100 % достоверность передачи информации;

модем поставляется с программным пакетом «Комплинк» для IBM PC/XT/AT или «Компмодем» для ДВК-2/3/4;

пакет программ содержит перестраиваемую базу данных, совместимую с dBase III +;

гарантийное обслуживание — 3 года.

Как известно, телефонные линии стран — членов СЭВ не соответствуют международным стандартам, что нередко приводит к трудностям при выборе модема. Рекомендуемое устройство позволяет обеспечивать не только достоверность передачи, но и ее высокую скорость, удовлетворяет требованиям самого взыскательного пользователя.

Модемы можно приобрести по адресу:

Москва, Нахимовский просп., д. 30/43, магазин-салон «Приборы и вычислительная техника», тел. 125-09-36, 124-20-42.

Ленинград, Ленинградский просп., д. 148, магазин-салон, тел. 290-55-34.

Подробные справки по телефону в Москве: 353-68-58 (с 14 до 19 ч.).

Научно-технический кооператив «КОМП»

Ремонт импортной вычислительной и множительной техники.
Заправка тонером картриджей ксероксов и лазерных принтеров.
Перепрокраска лент матричных принтеров в черный и другие цвета.
Поставка ленты для всех типов принтеров.
Все услуги по льготным ценам, исполнение на месте.
Оплата наличными в рублях или в валюте.
Ждем заявок у телефона **353-68-58.**

И. ПАНЧЕНКОВ

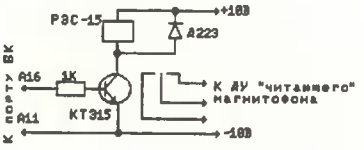
Приручение БК

Теперь рассмотрим проблему копирования программ. В ее решении наметились два подхода: программный и приборно-программный.

Первый, не требующий никаких дополнительных устройств, воплощен в примерно двух десятках программ-копировщиков. Среди них есть программы, целиком умещающиеся в стековой области БК и поэтому не теряющиеся при перезапусках БК; есть настолько «увешанные» сервисными возможностями, что их невозможно исчерпать в реальной работе. Для повседневной работы хороши ASCOP2 и HELP2. Первая копирует файлы не длиннее 37К*, вторая длину файлов не ограничивает, в остальном они очень похожи: обеспечивают переименование файлов, удвоенную плотность копирования, смещение адреса загрузки, изменение длины, фиктивное чтение. Копирование ведется через загрузку программ в экранную память, что позволяет визуально контролировать процесс — это весьма ценно.

Из приборно-программных способов лучше всего зарекомендовал себя предложенный Ю. А. Зальцманом (Алма-Ата). В его основе — использование простого согласующего устройства (рис. 6) и специально разработанного пакета программ-копировщиков, в котором наиболее интересна КОПИФН, позволяющая в автоматическом режиме копировать 32 файла, предварительно задав списком их имена. При прочтении копируемой ленты все «заказанные» файлы копируются дважды, остальные игнорируются. Недостатком является невозможность копирования с удвоенной плотностью и с изменением параметров программ (имен, адресов загрузки и т. п.). Полная автоматизация копирования, однако, искупает все недостатки.

Есть еще один способ копирования — простейший, но запрещенный инструкцией к БК: непосредственное переписывание программ с магнитофона на магнитофон. При хорошо отлаженной технике он дает приемлемый результат в 95 % случаев, а при использовании специального устройства — регенератора цифровых сигналов, разработанного П. Чирковым и Р. Булычом (описано в «ИНФО» № 5 за 1989 г.), обеспечивает качество копий, нередко превосходящее оригинал, и предельно автоматизирует процесс копирования любых программ, даже... защищенных от перезаписи!



6

К сожалению, на процесс копирования оказывает влияние такое количество мелких и, казалось бы, несущественных обстоятельств, что рассказать обо всех просто невозможно — лучше дать одну глобальную рекомендацию: чистите и смазывайте магнитофон как можно чаще (например, раз в неделю)!

Поскольку число сохраняемых программ, своих и чужих, имеет только одну тенденцию — к росту, довольно скоро перед обладателем БК встает проблема их каталогизации. Ее решают программы-каталогизаторы, формирующие перечни программ, записанных на кассету. Обычно эти списки включают названия файлов, адреса их загрузок, длины и общее число файлов на кассете.

Наиболее простые каталогизаторы фиксируют по мере считывания абсолютно все имеющиеся на кассете файлы; в результате получаются труднообозримые и неудобные для использования списки. Это, например, CATALOG и CATALOGM. В более удобных программах, примером которых является VERIFY2 А. Тереховского*, не только более продуман интерфейс пользователя, но и производится иллюстрирование процесса считывания файлов путем загрузки в экранную область.

К сожалению, VERIFY2 не подсчитывает

Окончание. Начало см.: Информатика и образование. 1990. № 4.

* Разработанная А. Соколовым, автором ASCOP2, программа ASCOP4 лишена этого недостатка.

* Ее описание опубликовано в «ИНФО» № 3 за 1990 г.— *Примеч. ред.*

контрольную сумму файла*, которая сегодня приобретает важное значение: с ее помощью можно опознавать авторские версии программ и, быть может, в дальнейшем удастся использовать ее для защиты прав авторов от посягательств фальсификаторов и несанкционированного копирования. Подсчет контрольной суммы предусмотрен в каталогизаторе ТСТФ Ю. А. Зальцмана, однако у него есть свои недостатки (например, восьмеричная запись номеров файлов, избыток незначащих нулей в цифровой информации). Одним словом, пользователь и здесь (как и вообще в области системных программ) находится в положении Агафьи Тихоновны из гоголевской «Женитьбы»: «Если бы губы Никанора Ивановича да приставить к носу Ивана Кузьмича...»

Теперь пришла пора заняться поисками нужного файла на магнитной ленте. Способом, ускоряющим эту весьма утомительную процедуру, является предварительная разметка ленты равноотстоящими друг от друга записями числовых отметок (маркерами), которые при считывании ленты воспроизводятся на экране, и запись файлов на размеченную ленту таким образом, чтобы перед началом каждого файла читался хотя бы один маркер, отыскать который не так уж трудно. Для разметки ленты можно воспользоваться такой, например, программой:

```
1.10 S A=1444
1.20 S A=A-2: I (FX(1,A)-2534)1.2,
                                  1.3,1.2
1.30 A "Включите магнитофон на запись
            и введите начальный номер
                              маркера" N
2.10 S M=FITR(N/100);S B=A+6;D 3
2.20 S M=N-M*100;S B=A+B;D 3
2.30 T %4.00,N;D 9;S N=N+1;G 2.1
3.10 X FX(-1,B,12336+256*M-
                        2559*FITR(M/100))
9.90 L S 0000
```

Неудобство этого способа в том, что разметка постепенно теряется, если поверх старых записей делать новые.

Более удобен способ, предложенный М. Медведевым (Москва). Он основан на использовании электронного счетчика оборотов приемного подкассетника и специальной программы DIR, которая позволяет создать в начале ленты файл-каталог всех программ, записанных на ленту, и соответствующих им показаний электронного счетчика. Пользуясь каталогом, можно выбрать нужный для чтения файл и под контролем DIR выйти на него в режиме ускоренной перемотки ленты вперед, после чего остановиться и загрузить файл в ОЗУ. Этот способ остроумен и эффективен; в какой-то степени он имитирует специальные лентопротяжные механизмы больших ЭВМ. В числе его недостатков — необходимость перемотки ленты в начало перед загрузкой каждого файла, но удобства все равно перевешивают.

В заключение стоит упомянуть, что для магнитофонов с хорошим обзором окна кассеты может быть рекомендовано предложенное А. Барсуковым (Минск) использование специальных измерительных линеек, наклеиваемых на кассету («ИНФО» № 3). При известном навыке этот метод дает примерно те же результаты, что и обычный механический счетчик метража ленты, так некстати отсутствующий в самых дешевых магнитофонах.

Впрочем, это нехитрое приспособление может быть встроено практически в любой магнитофон при наличии минимальных слесарных навыков, и сделавшие это не останутся внакладе.

Теперь настала пора сказать о том, как разумнее всего накапливать сведения о содержимом своих кассет. Эта проблема становится совсем нешуточной, когда число записанных программ начинает измеряться трехзначными числами. Как быстро отыскать нужную программу в бесконечных каталожных файлах? Как узнать об избыточности или, наоборот, недостаточности своей коллекции программ? Как сгруппировать имеющиеся программы? Такие или подобные вопросы постоянно встают перед каждым серьезным пользователем ПЭВМ.

Ответить на них позволяет программа BASE (авторы — А. Гречишкин, С. Фоменко, П. Успенский, известные под псевдонимом ASP согр., Зеленоград), формирующая так называемую базу данных, т. е. файл или систему файлов с произвольной организацией данных и доступа к ним. Именно в такой базе данных удобнее всего накапливать различные сведения, в том числе и об имеющихся программах. BASE позволяет классифицировать информацию по 22 (!) параметрам, связанным друг с другом так, что по любому из них можно отыскать совокупность всех остальных. Если учесть еще и продуманный сервис, предоставляемый BASE, становится ясно, что использование ее для упорядочения коллекции программ сулит много не только пользы, но и удобств. Ну а для более простых классификационных задач теми же авторами создана программа MBD (МиниБаза Данных), оперирующая только двумя параметрами. Она очень удобна для составления различных словарей, справочников и т. п.

* За время подготовки статьи к печати А. Тереховским разработан каталогизатор VERIFY2M, лишенный этого недостатка.

Создание, хранение и классификация информации — лишь один из элементов работы на ПЭВМ. Следующий неизбежный шаг — распространение, передача информации. «Штатный» для БК путь — использование магнитофонных кассет — все чаще дополняется использованием принтеров и плоттеров. По продуманности подключения внешних устройств БК не уступает лучшим компьютерам; к нему одинаково легко подключаются как неуклюжая отечественная, так и элегантная импортная периферия. Обеспечивает это хорошо продуманный порт ввода-вывода (64-контактный разъем на задней стенке компьютера).

Но как практически реализовать подключение всех внешних устройств к единственному разъему порта ввода-вывода? Здесь-то и пригодится разъем, вынутый из блока МСТД: он станет частью специального коммутационного устройства, построенного по принципу обычного розеточного тройника.

Но подключиться — еще не все. Каждое периферийное устройство работает под управлением своей специальной программы (эти программы называются драйверами). Иногда драйверы входят в состав более крупных программ — например, некоторые текстовые редакторы содержат драйверы принтеров, игровые программы — драйверы джойстиков.

Многообразие периферийных устройств порождает многообразие драйверов. Например, только за последние годы появились драйверы принтеров УВВПЧ 30-004, ТПУ15ВВВП80-002, ROBOTRON (ГДР), Think Jet (Hewlett-Packard, США), EPSON (США), DZM и D-100 (Польша); существуют драйверы и множества экзотических печатающих устройств — кассовых аппаратов, например, или почти антикварных электрических пишущих машинок «Консул»... Выбор драйверов огромен, а вот выбор принтеров сильно затруднен их дороговизной и дефицитностью.

Нелегко выбрать и оптимальный текстовый редактор для БК: нужно отдать предпочтение одной из 27 (!) программ. Впрочем, большинство пользователей БК «голосуют» за EDASP10, созданный уже упоминавшейся ASP corp. и улучшенный А. Шеломановым (г. Жуковский). Большой набор сервисных услуг этого редактора и формат записи обработанных им текстов считаются образцом для подражания, хотя мне кажется, что чрезмерность сервиса делает этот редактор громоздким и малоспособным справляться с основной задачей — созданием текстов максимальной длины с минимальными затратами времени и сил. Конечно, это дело вкуса, но не могу не противопоставить EDASPу редактор, включенный С. Кумандиным (Коломна) в состав ассемблера MIKRO10K (или MIKRO10KM для «высокой» клавиатуры), в котором оставлен только самый необходимый сервис и за счет этого в 2,5 раза увеличена длина обрабатываемого текста. И это преподносится пользователю между прочим, «в нагрузку» к ассемблеру! А очаровательная идея динамического распределения свободной памяти между текстом и буфером текста!..

Однако есть недостаток и у этого редактора — в нем отсутствует драйвер принтера, и в этом он проигрывает EDASPу, в котором драйвер есть. Есть драйвер принтера и в редакторе BED, а кроме того, в нем есть возможность форматировать текст, т. е. придавать ему типографский вид — выравнивать по правому краю, красиво разбивать на абзацы, центрировать; однако большого распространения BED почему-то не получил.

Обычно для форматирования используют программу ФОРМАТ2 А. Дьякова (Москва), позволяющую творить чудеса с текстами, написанными в формате EDASP (любой версии — от 0-й до 10-й). После обработки тексты приобретают чрезвычайно репрезентативный вид и могут быть распечатаны с помощью специального драйвера PRINT2 едва ли не на любом принтере, подключаемом к БК по параллельному способу ввода-вывода.

Комбинация EDASP, ФОРМАТ2 и PRINT2 вне конкуренции при обработке текстов, но эти программы пасуют, когда нужно работать с графической информацией — рисунками, схемами, диаграммами. Тут нужны так называемые графические редакторы, способные работать с изображениями так же легко, как текстовые редакторы с буквами. В основе подавляющего большинства их лежит классический принцип лупы, т. е. построение и редактирование рисунка в выделенной области экрана, где во внутренней системе координат воссоздается увеличенная в нужное число раз часть рисунка. Для примера можно назвать редакторы ГРЕД (0—4) (Д. Баранов), POINT (А. Ромашевский, Минск), ГРАТЕП (Л. Радченко), V003, GR30, GR30E, ГРАФРЕД. Из общего ряда заметно выделяется P01M (А. Бакерин, Москва), ориентированный на использование устройства координатного ввода типа «мышь» (УВК-01 производства ПО «Марсианка»). Он позволяет создавать многоцветные (до 24 псевдоцветов) изображения и пользоваться спецалфавитом, обладает хорошо продуманной системой сервиса, вплоть до выдачи изображения на принтер и записи на МЛ. P01M

знаменует переход к качественно новому программному обеспечению, рассчитанному на дальнейшее расширение возможностей БК-0010, и в первую очередь — памяти этого компьютера.

Надо сказать, что предпосылки к этому уже созданы: разработаны различные дополнительные платы памяти емкостью до 128К, эмуляторы электронных дисков, платы ПЗУ, хранящие наиболее важные и отработанные системные программы. А что вы скажете, если нам удастся в корпусе блока МСТД, который вы, надеюсь, сохранили, разместить дополнительное ОЗУ на 10 страниц по 16К байт?

Однако эта тема слишком сложна для того, чтобы обсуждать ее походя, и ей, я надеюсь, будут посвящены специальные публикации в следующих номерах «ИНФО».

Здесь же осталось рассмотреть вопросы, касающиеся работы на разных языках: ведь БК — сущий полиглот.

Сперва — о программах, ускоряющих ввод операторов Фокала и Бейсика. Это делается просто: некоторые клавиши (вплоть до почти всей клавиатуры) кодируются на ввод операторов и различных сочетаний символов. Из таких программ наибольшее распространение получили KEYBAS (С. Филатов, К. Гусев, Москва) для работы на MSX-Бейсике и FOKEY2 (А. Тереховский, Ленинград) для работы на Фокале. Для Фокала разработаны также программы FUNKEY и KEY1, являющиеся частными случаями более общей программы А. Тереховского.

Все эти программы позволяют записать выбранную кодировку клавиш на МЛ и при необходимости возвращаться к ней. Использование перекодированной клавиатуры резко повышает производительность труда при написании программ, сокращает время отладки.

Не всегда удается составить программу «на одном дыхании»; возникают фрагменты, вставки и т. п. Ни в Фокале, ни в Бейсике БК нет средств для сведения их воедино, но коллективный разум пользователей преодолел леность разработчиков: были созданы специальные программы сшивки. Для Фокала это СБОРКА и FOSBOR, а для Бейсика — MERGES и MERGE. Настоятельно рекомендую начинающим пользователям БК обзавестись этими прекрасно работающими программами: с ними не страшны никакие цейтноты, работу можно в любой момент прервать, можно начать с любого места, а уже затем скомпоновать получившиеся куски так, как надо.

В работе на Фокале есть еще один тупик, запланированный разработчиками: невозможность автоматической перенумерации строк программы. Эта операция часто бывает необходимой, и если в MSX-Бейсике она реализуется оператором RENUM, то в Фокале приходится использовать специальную программу ФОНОМ.

Что же еще сделано для БК? Очень многое! Например, созданы программы-трансляторы, переводящие тексты из форматов Бейсика и Фокала в классический формат EDASP и таким образом ликвидирующие ту или иную недостаточность заводской комплектации БК языками.

Созданы системные программы, расширяющие возможности Фокала, например XFOCAL (С. Зильберштейн, г. Киров), не только «подтягивающий» этот язык по возможностям к Бейсику, но даже делающий его в чем-то более мощным.

Созданы прекрасные отладчики программ (MIRAGE, OS0010F и т. д.).

Специальные программы МАШКОД и ФОКОД позволяют заниматься смешанным программированием, вставляя в Фокал-программы куски, написанные в кодах.

Создаются программы, позволяющие производить последовательную загрузку и выполнение нескольких программ под контролем самого БК, причем с сохранением промежуточных результатов в выделенной области памяти (COM, COMM.ASC, COMK.BIN*).

Усилиями в основном ленинградцев БК снабжен теперь еще языками Форт и Си. Сегодня известны ленточные версии языка Форт под названиями FORTH1.0, FIGFORTH и FORTH-83/8K с документационными файлами FORTHBOOK1—3 (или FORTDOK1—3) и редактором ЭДИТФ, а также ленточная версия языка Си, созданная А. Цаплевым (Ленинград) с документационным файлом DOCEDIT0 и несколькими тестирующими программами.

И наконец, с появлением систем, полностью управляющих магнитофоном (перемотка вперед и назад, включение режимов записи и воспроизведения), БК, несмотря на отсутствие дисковода и малое ОЗУ, становится универсальным компьютером, способным выполнять практически любую работу.

Конечно, в одной статье о многом рассказать не удалось. Я хотел дать лишь первоначальную ориентацию в сложном, противоречивом и пока плохо информационно организованном окружении БК-0010, ставшего самым распространенным компьютером в нашей стране. Надеюсь, мне удалось немного развеять туман предубежденности и плохой информированности, окружающий этот весьма скромный по замыслу, но тем не менее таящий в себе весьма серьезные возможности компьютер.

Ю. ЗАЛЬЦМАН

Архитектура и ассемблер БК-0010

Архитектура БК-0010

Войдем в наш «город» — БК-0010. В нем всего одна «улица», зато какая! В 65 536 «домов»! Это и есть адресное пространство нашей ЭВМ — 65536*д* байт. А что такое байт? Это блок информации, обычно состоящий из восьми двоичных разрядов, или битов. Самое малое число, которое может быть записано в один байт, это, конечно, ноль (000), самое большое — 255*д* (377). Переведем последнее число в двоичный формат. Что получилось? Правильно, восемь единиц — 11111111.

В БК нумерация «домов» имеет особенность. Они как бы стоят в два ряда. Не правда ли, вы и в своем городе такое встречали? Дом 56 и 56-а, 78 и 78-1. В БК каждый «дом» имеет свой номер, но четные «дома» стоят в «первой линии». Если каждый «дом» — это байт, то два байта называются *машинным словом*, или просто *словом*. Четные байты имеют, конечно, и четные номера — 0, 2, ..., 177776, причем номера четного байта и слова, к которому он принадлежит, совпадают. А нечетные? Они — «на задворках», и их номер на 1 больше, чем номер слова. Еще четный байт называют *младшим*, а нечетный — *старшим* байтом слова, в соответствии с их номерами. Зачем так сделано? Затем, что во многих случаях мы обращаемся сразу к двум соседним байтам, значит, к слову, и для этого байты объединены в слова. Но бывает, что нужен и более точный адрес — байт. К нечетному обратиться легко: его адрес отличается от адреса любого слова; а как быть с четным, адрес которого совпадает с адресом слова? Просто — ЭВМ для обращения к байту, в отличие от обращения к слову, имеет специальные команды, они-то и определяют, что нам нужно не слово, а четный байт.

Кстати, еще особенность — видели вы где-нибудь номер дома или квартиры «0»? А в ЭВМ везде и всюду ноль — такое же равноправное число, как любое другое, и даже более — нумерация всегда начинается с него!

Тут мы заметим, что каждый из разрядов (битов) байта или слова (в слове их, ясное дело, вдвое больше, чем в байте,— 16) тоже имеет свой номер. Нумерация их идет справа налево, и, как принято, начиная с 0. Правый, или младший, бит слова имеет номер 00, левый, или старший, бит — номер 15*д* (для битов, как ни странно, принята десятичная нумерация — традиция, а кроме того, по определенным соображениям это удобно). Можно сказать, что номера битов — это номера «квартир» в наших «домах» — байтах, причем в «домах второй линии» (нечетных байтах) они имеют двойную нумерацию — сквозную для слова (08—15*д*) и собственную для байта (0—7).

Итак, наш «город» из одной «улицы» имеет 100 000 слов, или 200 000 байтов, от байта 0 до байта 177777. А есть ли в этом «городе» районы? А как же!

Распределение адресного пространства

Давным-давно, в незапамятные времена (лет 30—40 назад), когда ЭВМ только-только появились, а «динозавры» — механические счетные машины еще не «вымерли», у первых ЭВМ память была раздельная — в определенных ее ячейках хранилась программа, а в других — данные, т. е. обрабатываемая информация. Позже от такой организации отказались, и теперь во всех универсальных ЭВМ программа и данные могут занимать любые ячейки памяти во всем адресном пространстве. И все же отдельные адреса адресного пространства (или, говоря проще, но менее строго, *адреса памяти*) имеют свое особое предназначение. Начнем с нулевого адреса БК и посмотрим, каково распределение адресного пространства.

Адреса 0—777 — *системная область и стек*. Это специально выделенные зоны ОЗУ, используемые чаще всего самим компьютером; более подробно об этом мы поговорим позже.

Адреса 1000—37 777 — *ОЗУ пользователя*. Здесь можно размещать программу и данные — что хотите и как хотите, это исключительное право программиста.

Адреса 40 000—77 777 — *экранное ОЗУ*. Мы уже говорили, что оно входит в общее адресное пространство и никак от него не отличается. Никак? Но ведь это экран дисплея и записанная туда информация может быть в любой момент разрушена. Стоит нажать любую клавишу — на экране появится символ, т. е. информация в зоне экранного ОЗУ изменится, а если нажать клавишу **СБР**, ин-

Продолжение. Начало см.: Информатика и образование. 1990. № 4.

формация будет мгновенно уничтожена (экран очищен). Используя экранное ОЗУ, мы всегда должны помнить об этом.

Взглянем еще раз на уже известные зоны ОЗУ. Экран занимает так много! Нельзя ли все-таки использовать хоть часть этой памяти? В БК-0010 это предусмотрено. Если нажать клавишу **СБР** по нижнему регистру (**АР2** в БК-0010.01), то экран «сожмется» до размера четырех строк текста, а программист получит в свое распоряжение дополнительную память, «отрезанную» от экранной (такой режим называется режимом расширения памяти, или РП). При этом распределение адресного пространства меняется: 1000—67 777—ОЗУ пользователя, а 70 000—77 777 — экранное ОЗУ. Заметим в скобках, что при переходе в РП та часть экранного ОЗУ, которая «прирезается» к ОЗУ пользователя, сохраняет информацию, хотя на экране ее уже не видно. Напротив, при обратном переходе к большому экрану экран очищается и информация, которая была до этого записана по адресам 40 000—67 777, безвозвратно теряется.

Следующий большой «район» нашего «города» начинается с адреса 100 000 и тянется почти до конца, по адрес 177 577. Это область ПЗУ. Каковы ее особенности? Если в область ОЗУ вы можете записать любую информацию по любому адресу (к чему это приведет — другой вопрос), то из области ПЗУ вы можете информацию только читать, на запись по этим адресам наложен запрет. Но тогда откуда же взялась та информация, которую мы оттуда читаем? Из специальных микросхем, в которые информация записана (или, как говорят, «зашита») на заводе-изготовителе. Изменить ее нельзя никакими средствами, доступными программисту. Это как бы наследственная память нашей ЭВМ, то, что она знает от рождения. Что же это за знания и как они размещены?

100 000—117 777 — *монитор-драйверная система* (МДС). Это, без преувеличения, важнейшая часть ЭВМ, без которой она превращается в мертвое железо. Здесь записаны все «основные инстинкты» ЭВМ. Ни одна операция, включая даже ввод символа с клавиатуры и вывод его на экран, не обходится без МДС. Это часть ПЗУ так тесно связана в обеспечении работоспособности ЭВМ с процессором и содержит столь важные программы функционирования всех систем машины, что ее принято называть виртуальным процессором (мы не станем расшифровывать этот термин).

Далее идет уже, так сказать, кора головного мозга. Без МДС машина мертва, но только с МДС она тоже немногое может — загрузить программу, запустить ее... Ни десятичной арифметики, ни чего-либо иного машина не знает. Она даже не может дать нам возможность просмотреть ее собственную память. Вся надежда на программы, загружаемые в ОЗУ! Но у машины есть и другие ПЗУ, причем они отличаются для различных типов БК-0010 (в последних моделях БК-0010.01 они есть всех типов и могут просто переключаться).

120 000—137 777 — интерпретатор языка Бейсик или языка Фокал.

140 000—157 777 — интерпретатор языка Бейсик или резервное адресное пространство (пустое место).

160 000—177 577 — интерпретатор языка Бейсик или *мониторная система диагностики и тестов* (МСД).

Уточним, что по техническим причинам одно ПЗУ БК-0010 включает не более 20 000 байтов.

Теперь вам ясно, что интерпретатор Фокала занимает одну микросхему ПЗУ, а Бейсика — три микросхемы. Ясно также, что Фокал не может работать одновременно с Бейсиком, а Бейсик — с МСД, так как они занимают одни и те же адреса. Что такое языки высокого уровня Фокал и Бейсик, вы, конечно, знаете. Но что такое МСД, какова ее роль?

Мониторная система диагностики (МСД)

МСД начинается с системы тестов нашей микро-ЭВМ. ЭВМ настолько сложное устройство, что проверить ее может только она сама! Поэтому и созданы специальные тест-программы, с которыми, конечно, вы знакомы — ведь ваш БК-0010 вы обязательно проверяли, согласно инструкции, как при покупке, так и в дальнейшем. Напомним, как это делается. После включения нажмите клавиши **ЛАТ** и **ЗАГЛ** (если у вас БК-0010.01, то предварительно должен быть подключен блок МСТД), а затем нажимайте последовательно: **Р, ПРОБЕЛ, Т, ВВОД.** На экране появится знак диалога тест-системы «+». Теперь в зависимости от того какой из тестов вы хотите провести, нажмите одну из цифровых клавиш 1—5 и выполняйте указания БК. Набор из пяти тестов проверяет все системы микро-ЭВМ исчерпывающим образом.

Из тест-системы можно перейти в основную часть МСД. Для этого в ответ на приглашение «+» нажмите клавиши **РУС, Т, С.** На экране появится знак диалога МСД — «¤». Это так называемый *знак денежной единицы*, или, на программистском жаргоне, «колесо» (иногда его называют также «черепашка», «солнышко» и т. д.). МСД, как и пусковой монитор (выход в который из Фока-

ла — **Р, ПРОБЕЛ, М, ВВОД**; из Бейсика — **MON, ВВОД**; знак диалога — «?»), имеет ряд команд, или директив, но количество их и, соответственно, возможности МСД намного больше. Отметим, что МСД называют еще *системой отладки* программ в кодах, что само по себе кое о чем говорит. Что же может МСД? Удобнее всего отвечать на этот вопрос, заодно изучая ее директивы.

Директивы МСД

МСД «понимает» директивы, состоящие из символов (в том числе букв в русском заглавном регистре, за одним исключением) и цифр, причем если в составе директивы присутствуют цифры, то общее правило таково, что цифры, следующие до буквы (или символа), задают какой-либо параметр. Буква (или иной символ), следующая без предшествующих цифр, параметр не задает, а индицирует, т. е. выводит на экран какое-либо значение. Помимо директив работы с памятью ЭВМ в набор команд МСД входят и директивы работы с магнитофоном. Все цифровые данные МСД принимает и индицирует только в восьмиричной системе счисления. Эти цифровые данные мы будем обозначать как α. Они могут состоять максимум из шести восьмиричных цифр, причем если цифр введено больше шести, то действительны только последние шесть. Незначащие нули впереди числа можно не набирать. Ошибку можно исправить с помощью клавиши **ЗАБОЙ** (перемещение курсора влево со стиранием символа).

αА — установить значение текущего адреса. Если мы введем, например, 1000А, то текущий адрес установится на 1000.

А — проконтролировать значение текущего адреса. МСД выдает ответ «=»; например, в нашем случае: А=1000.

αД — установить длину массива, байтов; Д — проконтролировать длину массива. Выполняется аналогично директиве А.

αР — размножить число α в диапазоне адресов, т. е. записать число α во все слова, начиная с адреса А до адреса А+Д. К примеру, если нам нужно обнулить (попросту говоря, стереть, или очистить) участок памяти, начиная с адреса 1000 до адреса 4000, мы набираем: 1000А3000Д0Р.

Х — подсчитать контрольную сумму массива, размещенного в памяти, начиная с адреса А до адреса А+Д. МСД выдает ответ «=». Например, подсчитаем контрольную сумму ПЗУ МДС: 100000А20000ДХ. Ответ: =177777, т. е. то же самое значение, которое БК выдает для первого ПЗУ, когда мы проводим ТЕСТ 1. Тут, может быть, уместно пояснить, что это такое. *Циклическая контрольная сумма* вычисляется сложением значений всех слов (в других случаях — байтов) контролируемого массива информации с прибавлением битов переноса за пределы слова (байта). Что такое, в свою очередь, перенос, станет ясно в дальнейшем (правда, еще не скоро, но не будем забегать вперед). Контрольная сумма позволяет убедиться в идентичности массивов, не сравнивая их целиком, так как вероятность ее случайного совпадения в данном случае не превышает $1/65536_д$.

αП — переслать по адресу α массив, расположенный, начиная с адреса А до адреса А+Д. Отметим, что «переслать» не совсем удачный термин, так как исходный массив тоже сохраняется в памяти. Правильнее было бы сказать «скопировать». Попробуем переслать содержимое какого-либо ПЗУ в экранную область памяти, это довольно эффектно выглядит. Выполним: 100000А 20000Д40000П. Содержимое ПЗУ почти мгновенно появляется на экране, что дает некоторое представление об истинном, т. е. не трансформированном языками высокого уровня, быстродействии ЭВМ — ведь ей пришлось переместить в памяти более 8000 чисел и проделать еще кое-какие операции! Пересылка возможна также и с пересечением зон адресов пересылаемых массивов, например 2000А10000Д1000П. Но пересылка в обратную сторону (1000А10000Д2000П) не даст ожидаемого результата — оба массива будут испорчены! Если вы учтете, что пересылка производится побайтно, начиная с начала массива, то легко догадаетесь, почему так происходит.

αС — сравнить в памяти два массива: эталонный, расположенный с адреса А до адреса А+Д, и контролируемый, расположенный с адреса α. В процессе проверки все расхождения информации в массивах выдаются на экран в формате:

	(эталонный массив)	(контролируемый массив)
(1)	адрес:данные	адрес:данные
(2)	адрес:данные	адрес:данные
	...	...
(*N*)	адрес:данные	адрес:данные

Если массивы полностью совпадают, на экран не выдается ничего. Например, попробуем переслать в память и сравнить с исходным содержимое того же ПЗУ: 100000А20000Д1000П — мы выполнили пересылку по адресу 1000. Теперь намеренно испортим массив в ОЗУ уже известной нам директивой: 2000А10Д0Р — мы записали в массив восемь байтов (т. е. четыре слова) нулей. Теперь выполним 100000А20000Д1000С. На экран будут выданы четыре строки указанного выше формата — «ошибка» в составе

контролируемого массива выявлена!

αЛ — листать (выдать на экран) массив начиная с адреса А длиной α байтов. После этого А принимает новое значение, равное А+α. Массив выдается на экран в виде машинных слов (а не байтов). Например, выполним 100000А10Л — на экран будет выдано содержимое первых четырех слов ПЗУ, после чего, если мы дадим директиву А, то получим: А=100010.

И — индикация содержимого слова по текущему адресу, αИ — запись числа α в слово по текущему адресу.

Б — индикация содержимого байта по текущему адресу, αБ — запись числа α в байт по текущему адресу.

Необходимо отметить, что как слово, так и байт выдаются в виде шести цифр. При выдаче байта, конечно, три старшие цифры всегда нули. При записи байта может быть введено с клавиатуры любое количество цифр, но запись байта производится только по модулю 256*д* (т. е. остатка от целочисленного деления на 256*д*) шести цифр, набранных последними.

Ц — циклическое (т. е. непрерывное) чтение слова по адресу А. Команда может быть полезна для исследования ячеек памяти с меняющейся во времени информацией. Выход из цикла — клавиша **СТОП.** Например, дайте директивы 177662АЦ, затем нажимайте различные клавиши и наблюдайте за изменением информации.

Щ — снять защиту системной области. Мы уже отмечали, что область адресов 0—777 называется системной и используется преимущественно для собственных нужд ЭВМ. Запись туда случайной информации чревата неприятностями, вплоть до полного отказа ЭВМ (конечно, временного, до отключения питания или перезапуска, но информация, хранящаяся в ОЗУ, будет утрачена). Чтобы избежать неприятностей, системная область защищена в МСД от случайной записи, и при попытке записать информацию по адресу 0—777 выдается сообщение ЗЩ — защита. После подачи директивы Щ защита снимается и возможна запись в системную область. Восстановление защиты — клавиша **СТОП.**

, (запятая) — чтение слова с *инкрементом* (инкремент — увеличение). По данной директиве текущий адрес А увеличивается на два, а затем содержимое ячейки по этому адресу выводится на экран. α, — запись слова по текущему адресу с инкрементом: в слово с текущим адресом записывается число α, а затем адрес увеличивается на два и выводится содержимое слова по этому адресу. Данные директивы используются преимущественно для чтения или записи последовательных ячеек (зон) памяти.

— (минус) — чтение слова по текущему адресу с *декрементом*, α— — запись числа α в слово по текущему адресу с декрементом (декремент — уменьшение). Данная директива аналогична предыдущей, с той лишь разницей, что значение текущего адреса А не увеличивается, а уменьшается.

. (точка) — чтение байта с инкрементом, α. — запись байта с инкрементом.

: (двоеточие) — чтение байта с декрементом, α: — запись байта с декрементом.

Эти две директивы аналогичны двум предыдущим, но работают не со словом, а с байтом, и увеличение или уменьшение текущего адреса при их исполнении происходит не на два, а на единицу.

МП — пуск мотора магнитофона (при наличии дистанционного управления (ДУ) от БК-0010). Применяется для дистанционного включения магнитофона. Останов мотора — любая символьная клавиша или клавиша **СТОП.**

МФ — фиктивное чтение файлов с магнитной ленты (МЛ). На запрос «ИМЯ=» нужно ввести имя того файла, после которого магнитофон должен быть остановлен. Загрузка файла в ОЗУ и проверка контрольной суммы по данной директиве не производятся, а имена, не совпадающие с заданным, выводятся на экран.

МЧ — загрузка файла с МЛ. На запрос «АДРЕС=» необходимо ввести адрес, начиная с которого должен быть загружен файл, а на запрос «ИМЯ=» — имя файла, который необходимо загрузить. Если вместо адреса задать число 0, то файл будет загружен по адресу, указанному в *оглавлении файла*, т. е. специальном блоке, записанном на МЛ в начале файла. Фактически файл при этом будет загружен в ту область памяти, из которой он был записан на МЛ, а чаще всего именно это и требуется.

МЗ — запись массива в виде файла на МЛ. На запрос «АДРЕС=» вводится адрес начала массива, на запрос «ДЛИНА=» — длина массива в байтах, на запрос «ИМЯ=» — имя файла, под которым он должен быть записан (имя файла в МСД состоит из 16*д* *любых* символов; если задано менее 16*д* символов, они автоматически дополняются до 16*д* символов пробелами). Все указанные параметры заносятся в оглавление файла при его записи на МЛ.

Заметим, что при исполнении трех последних директив магнитофон с помощью ДУ включается автоматически, а по окончании чтения или записи (или при нажатии клавиши **СТОП**) отключается. По окончании набора очередных цифровых данных или имени файла необходимо нажимать клавишу **ВВОД,**

исправлять любые данные можно клавишей **ЗАБОЙ** (←).

К — выход из МСД. При этом управление передается Фокалу, содержимое ОЗУ пользователя стирается (кроме зоны адресов 1000—1377).

ТК — выход в тесты. По данной директиве появляется знак диалога тест-системы «+». Если не проводить тесты 1 и 5, то содержимое ОЗУ сохраняется.

ТД — переход к тестам внешней диагностики. Применяется при наличии дополнительной диагностической аппаратуры.

Т0—Т5 — запуск тестов МСД. Эти тесты отличаются от тестов тест-системы. Вы можете провести их на своем БК и ознакомиться с ними, поэтому здесь нет необходимости их описывать. Отметим, что весьма полезным является тест 2 МСД.

αG — запуск программы по адресу α, т. е. передача управления по этому адресу. Это единственное исключение, когда директива МСД подается в регистре ЛАТ—ЗАГЛ.

Теперь, после ознакомления с директивами МСД, пора ответить на вопрос, для чего может быть полезна система отладки.

Ясно, что в МСД можно как просматривать память ЭВМ, так и изменять ее содержимое, т. е. заносить в ячейки памяти информацию. Зная *язык машинных кодов* (или машинных команд), в МСД можно писать программы в кодах, вводить массивы информации, записывать их в виде файлов на МЛ, запускать, отлаживать.

Хотя язык машинных кодов не так сложен, как принято обычно считать, писать программы непосредственно в кодах неудобно, это требует большого напряжения и отличной памяти. Поэтому разработан специальный вспомогательный *язык ассемблера,* позволяющий достичь тех же самых результатов гораздо меньшей ценой. Впрочем, имея готовую распечатку программы в кодах, можно относительно легко ввести ее в память, пользуясь директивами МСД. Это позволяет публиковать тексты сравнительно небольших программ в печати, избавляя от необходимости тиражировать их непосредственно на МЛ. Если вам требуется ввести такую программу, то, чтобы избежать ошибок, рекомендуется ввести ее в память ЭВМ дважды, по разным адресам, а затем провести сверху этих массивов — маловероятно, что в обоих случаях вы сделаете одинаковые ошибки. Попробуйте ввести в память и запустить простейшую программу в кодах, для чего выполните директивы:

```
1000A
12700, 14, 104016, 12700, 101, 104016, 0,
1000G
```

На выводимые по директиве «запятая» числа не обращайте внимания и набирайте новые. После запуска эта программа выполняет сброс экрана и выводит символ «А». Как видите, программирование в кодах — не такая уж сложная штука!

Директивы МСД для работы с магнитофоном, как легко догадаться, позволяют не только загружать программы с МЛ, но и копировать их, если после загрузки записать на МЛ загруженный массив, причем безразлично, на каком языке эта программа, нужно лишь знать ее адрес и длину.

В заключение приведем некоторые полезные сведения, которыми можно воспользоваться при работе в МСД.

Прочитать параметры успешно загруженного файла можно, если знать, что в ячейке 264 хранится адрес его загрузки, а в ячейке 266 — длина: достаточно дать директивы 264А4Л. Контрольная сумма файла хранится в ячейке 312 и выводится директивами 312АИ. Но можно прочитать параметры файла (кроме контрольной суммы), и не загружая его, для чего необходимо в режиме фиктивного чтения (МФ) прочитать имя файла (независимо от того, совпадает ли оно с заданным), после чего адрес файла окажется в ячейке 346, а длина — в ячейке 350, причем адрес файла будет «истинный», т. е. тот, который указан в его оглавлении на МЛ, а не заданный в процессе загрузки, как это имеет место при чтении ячейки 264. Ячейки 346 и 350 могут быть прочитаны и после загрузки файла с тем же результатом.

По директиве 100274G происходит переход в *пусковой монитор* (ПМ, символ диалога — «?») без стирания ОЗУ (ПМ мы не описывали, так как он практически всем хорошо известен и значительно уступает МСД по набору директив и своим возможностям). Перейти из ПМ в МСД можно через тесты, нажав клавиши **Т, ВВОД, РУС, Т, С** (также без стирания ОЗУ).

Перейти из МСД в Фокал без стирания ОЗУ можно, выполнив директивы

```
Щ
120020А26Д0П
262А177777И
СТОП
```

после чего на экране появляется сообщение Фокала «ОСТАНОВ ПО КЛАВИШЕ СТОП». Чтобы такой переход был успешным, выход из Фокала должен быть произведен непосредственно в МСД, а не через пусковой монитор, в МСД же не должны загружаться (а тем более запускаться) программы в кодах. При соблюдении этих условий после выхода в Фокал содержимое ОЗУ полностью сохраняется и могут даже запускаться для дальнейшей работы ранее загруженные в Фокале программы, возможен просмотр их

листинга и т. п. Переход Фокал — МСД — Фокал может делаться таким способом неоднократно, например, с целью модификации в МСД Фокал-программ или изучения их формата.

Выйти из пускового монитора в Бейсик без стирания ОЗУ (с сохранением ранее загруженной Бейсик-программы) можно командой **С120234 ВВОД.** Правда, ключи при этом не сохраняются.

Диагностические сообщения МСД

В процессе работы в МСД система ведет развернутый диалог с пользователем, выдавая различные указания и сообщая результаты исполнения директив. Обычно сообщения МСД выдаются в развернутой форме и не требуют особых комментариев.

Но есть несколько специальных диагностических сообщений МСД, которые весьма кратки и нуждаются в пояснении. С одним из таких сообщений мы уже познакомились: ЗЩ — защита системной области. Выдача этого сообщения означает, что произведена попытка выполнить с помощью одной из директив МСД запись в область адресов 0—777. Напомним, что на запись в системную область наложен запрет (до снятия защиты) исключительно для директив МСД, но не для машинных команд, поэтому от программы пользователя системная область никак не защищена и ее защита от записи неправильной информации — забота программиста.

Еще два сообщения МСД нам пока не встречались, и они нуждаются в более подробном рассмотрении.

ЗВα — зависание по адресу α. Данное сообщение выдается в двух основных случаях: попытка записи по адресу ПЗУ; попытка чтения или записи по несуществующему адресу.

Относительно записи информации в область адресов, занимаемую ПЗУ, все ясно — запись по адресам 100000—177577 запрещена аппаратно и невозможна по техническим причинам. А что такое несуществующий адрес? Это адрес, где нет реального аппаратного устройства, которое могло бы ответить на вызов процессора. При выполнении подобных некорректных операций процессор, не получив ответа от устройства (например, ОЗУ), выполняет так называемое прерывание по зависанию и производит переход по вектору 4 (что это такое, станет ясно в дальнейшем), результатом чего и является выдача сообщения ЗВ (зависание) и значения текущего адреса А. Понятно, что если зависание возникло в результате исполнения директивы МСД (например, 100000А377И), то выданное значение обычно и есть тот адрес, по которому произошло зависание. Но в прочих случаях, например при исполнении программы, выданное значение адреса и адрес, по которому произведено некорректное обращение, как правило, не имеют ничего общего. В этом случае информацию несет только само сообщение ЗВ.

НК — неправильная команда. В набор команд процессора входит строго фиксированный перечень кодов. В случае, если код очередной команды, встретившейся в программе, не входит в этот перечень, процессор выполняет *прерывание по резервному коду* и производит *переход по вектору 10* (что также будет в дальнейшем разъяснено). В результате и выдается сообщение НК. Следует особо отметить, что далеко не весь набор команд процессора К1801ВМ1 использован в БК-0010 (или, как говорят, реализован аппаратно). Вследствие этого есть такие коды (не перечисленные, в частности, в «Руководстве системного программиста», прилагаемом к БК), которые вместо ожидаемого сообщения НК вызывают сообщение ЗВ. Примером является, например, код 12. По этому коду процессор должен был бы произвести запись некоторой информации в область системных регистров. Но нужных регистров в адресном пространстве БК-0010 просто нет, поэтому и происходит прерывание по зависанию. Примером же действительно несуществующего кода команды является, например, код 30, вызывающий, как и положено, сообщение НК. Знание этих особенностей поведения процессора позволяет иногда разобраться в загадочных, казалось бы, сообщениях при отладке программы.

Необходимо отметить, что описанные сообщения выдаются только в том случае, если выполняются программы пользователя, загруженные и запущенные в МСД (либо выполняются директивы самой МСД). Если же загрузка и запуск программ производятся из ПМ либо программа пользователя изменяет значение векторов прерывания 4 и 10, диагностические сообщения МСД, разумеется, выдаваться не будут.

Итак, мы вкратце познакомились с мониторной системой диагностики МСД, хотя нам и пришлось ради этого надолго прервать описание архитектуры ЭВМ. Но затраченное время в дальнейшем окупится — теперь вы сможете не просто знакомиться с материалом, предлагаемым вашему вниманию, но во многих случаях активно его проверять на своем БК, экспериментировать. Никогда не упускайте случая потренироваться в работе с МСД, а заодно и проверить автора — он ведь тоже человек и может ошибаться!

Продолжение следует

Работа с ассемблером. Формат команды

Итак, загрузив и запустив ассемблер-систему МИКРО.10К с адреса 1 000, войдем в редактор по команде EN. Она обеспечивает переход в начало текста, а если нам нужно вернуться на ту страницу, на которой мы были в момент выхода из редактора, дается команда ED. Теперь можно писать текст программы, состоящий из *команд ассемблера*.

В каждой строке текста программы записывается одна команда (исключением являются *псевдокоманды*, которых в каждой строке может быть несколько, но об этом позже). *Командная строка* текста программы условно разделена на четыре *поля*. Слева направо расположены поля *метки*, *оператора*, *операндов* и *комментариев*. Поля в редакторе МИКРО.10К отделяются друг от друга любым числом символов **ПРОБЕЛ** (код 40) или **ГТ** (код 11). Удобнее всего разделять поля нажатием клавиши **ГТ**, при этом одноименные поля размещаются друг под другом, программа имеет аккуратный вид и легко читается. Оканчивается строка символом **ВВОД** (код 12). Рассмотрим каждое из полей командной строки.

Первое поле — **поле метки.** Метка — это условное символьное имя, присваиваемое данной строке. После трансляции каждая строка текста превращается в последовательность кодов, размещаемую начиная с определенного *текущего адреса*. Метке, стоящей в начале строки, присваивается этот адрес, который заносится в формируемую транслятором *таблицу меток*. Зачем нужны метки? Как вы помните, в Бейсике или Фокале каждой строке присвоен номер, пользуясь которым можно к ней обратиться (передать ей управление) из любой точки программы. Точно так же в ассемблере можно обратиться к данной строке (оператору), передав ему управление по имени метки. Но есть и отличия метки от номера строки.

Во-первых, поле метки не является обязательным, т. е. помечаются не все строки, а только те, к которым придется обращаться по ходу исполнения программы. Во-вторых, метками могут помечаться не только исполняемые строки программы (содержащие операторы), но и любые другие адреса (ячейки памяти). Таким образом, по метке можно не только передать управление команде, но и просто обратиться к любому заданному адресу, например записать в помеченную ячейку памяти какое-либо значение или прочитать из нее число. Следовательно, метка — это еще и имя переменной или константы. С другой стороны, как передать управление, так и обратиться к ячейке памяти можно и по адресу. Метки же, в общем случае, служат для присваивания отдельным адресам имен и значительно облегчают процесс программирования на ассемблере.

Метка может состоять из латинских заглавных букв и цифр, причем допустимое количество символов в имени метки — не более трех (в МИКРО.1103 — не более шести). Если метка начинается с буквы, она называется *обычной* и обращение к ней возможно из любого места программы, с помощью любого оператора, имеющего поле операндов (в частности, передающего управление). Если же метка начинается с цифры, она называется *локальной*, обращение к ней возможно с помощью только некоторых операндов и она действительна до ближайшей обычной метки (вперед или назад). Имя обычной метки, как правило, должно быть для данной программы уникальным, т. е. его повторение не практикуется (при повторении имени имеет смысл только последняя из одноименных меток). Имена локальных меток могут по ходу программы повторяться многократно, лишь бы одноименные локальные метки были разделены обычными (в ранних версиях ассемблеров допускалось употребление только обычных меток). Для отличия имен меток от прочих имен после метки ставится символ «:». Имена меток и операторов могут совпадать. Если в строке имеется несколько меток, то всем им присваивается один и тот же адрес.

Приведем примеры. Обычные метки:

```
A:    AB:    MET:    N9:    T75:    TR4:
```

Локальные метки:

```
2:    94:    545:    7N:    4A7:    BAR:
```

Запись нескольких меток в одной строке:

```
LD:M1:    1:12:    A:17:14:
```

Второе поле командной строки — **поле оператора.** Оно является обязательным, без него строка лишена смысла. Оператором называется собственно машинная команда, записанная на языке ассемблера. Набор операторов в языке ассемблера соответствует набору машинных команд, исполняемых ЭВМ. Таким образом, оператор сообщает процессору, что надо сделать. Подробно каждый из операторов будет рассмотрен в дальнейшем, пока же опишем только два, которые будем использовать в приводимых примерах.

CLR — *очистка* (обнуление) операнда (заданной тем или иным способом ячейки памяти или регистра).

MOV — *пересылка* содержимого первого операнда во второй (т. е. перезапись содержимого одной ячейки памяти в другую; при этой операции содержимое первой ячейки не меняется, поэтому название «пересылка» следует признать не совсем удачным, более

точным было бы название «копирование»).

В состав оператора входит также указание, работает он со словом или с байтом. Если имя оператора обычное, то это указывает на работу со словом; для работы с байтом к имени оператора прибавляется буква B, например: CLR — CLRB, MOV — MOVB. Имена операторов в ассемблере записываются латинскими заглавными буквами.

Третье поле командной строки — **поле операндов**. В зависимости от стоящего в строке оператора операндов может быть два, один, ни одного (т. е. поле операндов не является обязательным). Если оператор указывает процессору, что ему надо сделать, то операнды — указание на то, с чем это надо сделать, т. е. операнды играют роль переменных или констант. Если поле операндов отсутствует, то либо выполняются действия над подразумеваемыми, строго фиксированными для данного оператора операндами (например, операторы переходов заносят адрес перехода в счетчик команд), либо команда является управляющей, т. е. изменяет состояние ЭВМ или порядок выполнения программы.

Способы указания операндов весьма разнообразны, они носят название *способов адресации* и будут подробно рассмотрены ниже.

Последнее поле командной строки — **поле комментариев**. Как и в других языках, комментариями называются пояснения к программе, используемые с целью сделать текст программы более понятным. Транслятор игнорирует комментарии, они сохраняются только в листинге программы, а в загрузочный модуль (программу в кодах) не включаются, поэтому в состав комментариев могут входить любые символы. Комментарии могут следовать после оператора или операндов без всяких разделителей, транслятор распознает их просто по месту расположения в строке. Если же комментарии записаны в пустой строке либо сразу после метки, они должны начинаться с символа «;». В ранних версиях ассемблеров отделение комментариев символом «;» было обязательно во всех случаях.

Приведем примеры записи полных командных строк ассемблера и комментариев.

```
Метки Опе- Операнды  Комментарии
      раторы

; Программа начальной очистки счетчиков
        ;              Очистка счетчика
BEG:    CLR    M1      единиц
        CLR    M2      десятков
        CLR    M3      сотен
OST:    MOVB   #1,PRO  Запись признака
        ;              останова в ячейку
        ;              признака
CON:    ;            Продолжение программы
        MOV    M1,M4   Запись единиц
; и т.д. - остальной текст программы
```

Из-за типографских ограничений листинги программ приведены не в том виде, какой они имеют в редакторе ассемблера (на самом деле в строке 64 символа), за что автор и редакция приносят читателям свои извинения.

Разумеется, многое в этой записи вам еще непонятно, но все станет ясно в дальнейшем. Пока же обратите внимание на расположение полей меток, операторов и операндов, на правила записи комментариев.

В общих чертах вы теперь знаете, как писать текст программы. А что с ним делать потом? Разумеется, в свое время об этом будет рассказано подробно. Но по ходу изложения будут приводиться примеры, которые многие читатели захотят проверить, а может быть, и видоизменить, чтобы посмотреть, что получится. Такие эксперименты можно только приветствовать, для освоения ассемблера (как и любого языка программирования) нужна практика. Но столкнувшиеся с ассемблером впервые наверняка испытают трудности при решении вопроса: что делать с написанным текстом программы? Поэтому, забегая вперед, дадим самые общие и схематичные рекомендации, не объясняя пока, «что для чего и зачем». Рекомендации строго конкретны и относятся только к ассемблер-системе МИКРО.10К. Если у вас другая ассемблер-система, найдите по ее описанию эквивалентные директивы самостоятельно.

Итак, вы набрали в редакторе текст программы. В конце поставьте псевдооператор END (в отдельной строке). Все, что следует за ним, транслятор игнорирует. Теперь нужно выйти из редактора, нажав клавишу **КТ**. Вы в мониторе. Если хотите, запишите текст программы на МЛ директивой ST. Оттранслируйте текст директивой CO, по окончании трансляции будет выдана длина загрузочного модуля в байтах. Если транслятор обнаружит ошибку, то на запрос E/C? нажмите клавишу **Е**, транслятор передаст управление в редактор, и курсор покажет ошибку, «ткнет в нее носом». Частая ошибка, которую трудно заметить начинающему,— случайное употребление русского символа вместо латинского, одинакового по начертанию.

Текст оттранслирован. Определим, нужна ли компоновка. Если все метки (имена и адреса) транслятор выдал в обычной форме, компоновка не нужна. Если же есть метки, выданные инверсно, то нужна. Для компоновки дайте директиву LS; последует запрос «Addr=». Введите число 13 000 и нажмите **ВВОД**. Во вторично выданной таблице инверсных меток быть не должно. Если они

есть, значит, в программе ошибки (мы пока даем указания, касающиеся только приводимых далее учебных программ, при реальном программировании все несколько сложнее). Если все в порядке, программу в кодах (загрузочный модуль) можно записать на МЛ директивой SA и/или запустить директивой RU. Посмотрев, как работает программа, остановите ее клавишей **СТОП**; ЭВМ выйдет в МСД. Вы можете с помощью директив МСД просмотреть программу в кодах; ваш загрузочный модуль расположен начиная с адреса 13 000. Можете, если хотите, еще раз запустить программу с адреса 13 000. Если захотите внести в программу изменения, войдите в ассемблер по «холодному входу» — 1002G, затем в редактор — ED или EN и редактируйте текст. Если же вы захотите текст уничтожить и начать все сначала, то либо запустите ассемблер с адреса 1000, либо в мониторе дайте директиву RS. Текст с МЛ загружается директивой LO, а если желательно при этом сохранить и прежний текст в памяти, дайте директиву LF. Остальные директивы монитора вам не понадобятся еще долго (а если вы не освоите ассемблер всерьез, то вообще никогда).

Вот и все. Как вы считаете, это настолько уж сложнее Фокала или Бейсика, чтобы стоило об этом говорить? А куда как часто автору приходилось встречать умных (в остальных вопросах) людей, которые заявляли, что на ассемблере ОЧЕНЬ трудно работать, что он УЖАСНО сложен и что освоить его могут разве лишь гениальные люди... Скромность хороша, но не тогда, когда она мешает работать.

Несколько слов о редакторе текста. Если вам приходилось работать с редактором EDASP, то осваивать в редакторе МИКРО.10К почти ничего не придется, они очень похожи. А когда вы полностью освоите МИКРО.10К, у вас будут основания забыть о всех других редакторах текста.

Способы адресации

Этот раздел, видимо, самый сложный. Но он важнейший в изучении ассемблера, ибо знание его необходимо для правильного использования операторов; потому на его изучение не следует жалеть сил и времени.

Итак, в команде ассемблера *оператор* — это указание, *что надо сделать*, или собственно команда, а *операнд* — это указание, *с чем надо сделать*. Как уже было сказано, поле операндов не обязательно, т. е. присутствует не во всех командах. Но если его нет, это значит только, что данный оператор в операндах не нуждается: либо они подразумеваются, либо он обращается не к памяти ЭВМ, а к иным ее устройствам. В большинстве случаев операнды все же присутствуют в составе команды, и разговор пойдет именно о них.

Способы (или *методы*) *адресации* не что иное, как способы указания на те или иные ячейки памяти, с которыми должен манипулировать оператор. Этих методов в нашей системе команд весьма много, что обеспечивает высокую гибкость в построении программы и является большим преимуществом системы команд данного типа ЭВМ. Например, многие способы адресации не только указывают на адреса памяти, но и модифицируют эти адреса, т. е. по сути, выполняют функции дополнительных команд, а следовательно, экономят память и повышают скорость исполнения программы. Способы адресации практически одинаковы для всех команд, в которых присутствуют операнды. Есть некоторые ограничения, относящиеся к отдельным операторам, но их мы будем оговаривать особо, когда дойдем до описания этих операторов.

Прежде чем перейти к изложению конкретных способов адресации, кратко напомним, что в нашей ЭВМ вообще можно адресовать, т. е. какие возможны операнды.

Прежде всего это ячейки памяти, лежащие в пределах адресного пространства ЭВМ, т. е. по адресам 0—177777. Ячейку можно адресовать с точностью до слова или до байта, но это уже дело формы записи оператора, а не операндов.

Адресовать можно также регистры общего назначения процессора (РОН). Их восемь, и они носят названия R0, R1, R2, R3, R4, R5, R6 и R7. Но на языке ассемблера регистр R6 принято именовать SP, а R7 — PC. Подробно эти моменты рассматриваются в разделах, посвященных архитектуре ЭВМ, здесь же важно, что все это в принципе есть ячейки памяти и что они могут адресоваться, т. е. выступать в роли операндов. Напомним еще, что SP — это указатель вершины стека, а PC — счетчик команд, т. е. указатель адреса следующей команды.

Теперь можно перейти к описанию собственно способов адресации. Приводя примеры, мы будем стараться пользоваться только теми двумя операторами (CLR и MOV), с которыми мы познакомились. РОН «вообще» мы будем обозначать «RN», подразумевая, что это может быть любой из восьми РОН. Итак, приступим.

1. Регистровая адресация (иногда ее называют еще регистровой прямой в отличие от косвенной). При этом способе операндом является один из РОН. Записывается как RN. Примеры:

```
CLR    R2        Очистить (обнулить)
;                R2
MOV    R2,R4     Переслать (скопиро-
```

```
;               вать) содержимое R2
;               в R4
MOV    R4,SP    Переслать содер-
;               жимое R4 в SP
```

Этот способ адресации самый простой и в дальнейших пояснениях не нуждается.

2. Регистровая косвенная адресация. При этом способе в одном из РОН содержится не само число, с которым нужно работать, а его адрес, т. е. номер ячейки памяти, в которой число находится. Записывается как (RN) или @ RN. Что такое скобки, видимо, объяснять не требуется. Укажем только, что скобки в ассемблере всегда круглые; а вот со значком @ знакомы не все. Это так называемое *коммерческое эт*. Название это длинное и труднопроизносимое, поэтому программистский жаргон присвоил данному символу пусть не слишком строгое и благозвучное, но зато удобное наименование «собачка». Обозначения @RN и (RN) абсолютно эквивалентны, но более принято все же писать @RN, чтобы данный тип адресации отличался от адресации с инкрементом или декрементом, о которых речь дальше. Приведем примеры.

Пусть, например, в регистре R2 записано число 1 000, тогда команда

```
CLR    @R2
```

обнулит ячейку 1000, т. е. ту, адрес которой указан в R2.

Пусть в R2 записано число 1000, а по адресу 1000 — число 1254. Тогда команда запишет в R4 число 1254, ведь именно его адрес указан в R2.

```
MOV    @R2,R4
```

3. Автоинкрементная косвенная адресация. Этот вид адресации еще сложнее: помимо основного действия (косвенное обращение к ячейке памяти) происходит еще изменение (модификация) адреса этого обращения. Поясним, что *инкремент* — это увеличение чего-либо. В данном случае увеличивается указатель адреса ячейки памяти, к которой мы обращаемся, т. е. содержимое регистра, служащего указателем адреса. А *авто* — указание на то, что данное увеличение происходит само, не требует специальной команды. Записывается эта адресация как (RN)+. То, что знак + стоит после имени регистра, намекает на порядок выполнения команды: сначала делается нечто с ячейкой, на которую указывает адрес, помещенный в регистр RN, а потом уже содержимое регистра увеличивается... на сколько? В зависимости от формы оператора: если оператор работает с байтом (т. е. имеет в конце имени букву B), то на 1 (переход к адресу следующего байта), а если оператор работает со словом, то на 2 (переход к адресу следующего слова). Как видите, все логично. Но из последнего правила есть исключение: если в качестве указателя адреса используется регистр SP, т. е. адресуется ячейка стека, то автоинкремент делается только на 2 независимо от формы оператора. Приведем примеры.

Пусть в регистре R2 — число 1000. Тогда команда

```
CLR    (R2)+
```

очистит слово по адресу 1000, а содержимое регистра R2 станет равно 1002. Если затем выполнить команду:

```
CLRB   (R2)+
```

то очистится байт по адресу 1002, а содержимое R2 станет равно 1003.

Данный способ адресации применяется обычно для работы с массивами. Если записать в регистр начальный адрес массива, а затем выполнить команду с автоинкрементной адресацией столько раз, сколько в массиве элементов, то команда (одна и та же, выполняемая, например, в цикле) каждый раз будет работать со следующим элементом массива: обнулять его или пересылать куда-либо и т. п. Если оператор имеет два операнда, как, например, MOV, то автоинкрементная адресация работает с каждым из них как с отдельным. Пусть в R2 число 1000, а в R4 — 2000. Команда

```
MOV    (R2)+,(R4)+
```

перешлет содержимое ячейки 1000 в ячейку 2000, после чего числа в R2 и R4 будут равны соответственно 1002 и 2002.

Особый «хитрый» способ — автоинкремент регистра PC. Пусть, например, после команды записана константа или переменная и ее надо «извлечь». Запишем:

```
MOV    (PC)+,R4
  <константа>
```

Что произойдет? Число КОНСТАНТА запишется в регистр R4, ведь в момент исполнения команды в PC находится адрес следующей команды, т. е. числа КОНСТАНТА! А затем PC увеличится на 2, т. е. управление будет передано уже на следующую за константой команду.

4. Автоинкрементная двойная косвенная адресация. Этот способ не слишком отличается от предыдущего, только в регистре-указателе при этом не адрес числа, с которым должен работать оператор, а адрес адреса. Записывается как @(RN)+. Поясним, что это значит, примером. Пусть в ячейке 1000 записано число 1254, а в ячейке 1254—

число 3333. Тогда если в R2 находится число 1000, то команда

```
MOV   @(R2)+,R4
```

перешлет в R4 число 3333, а содержимое регистра R2 станет равным 1002. Данный способ применяется относительно редко, но тем не менее он очень ценен, так как позволяет работать, например, с таблицами. Пусть каждому элементу таблицы А приписан адрес определенного элемента таблицы Б. Данная адресация позволяет, двигаясь по адресам первой таблицы, например, проверять элементы второй. Причем особенно интересно, что если в первой таблице элементы следуют друг за другом, то элементы второй таблицы могут быть произвольно «рассеяны» по памяти, и это не усложнит работу, лишь бы элементы первой таблицы содержали адреса элементов второй.

Два следующих способа адресации сильно напоминают два предыдущих, хотя есть некоторые нюансы.

Продолжение следует

Калейдоскоп

Уважаемая редакция!

Мной разработаны схемы расширения ОЗУ БК «+32», «+128», «+256», не требующие «потрошения» внутренностей и сохраняющие аппаратную совместимость с существующим программным обеспечением. Расширение предоставляет пользователю адресное пространство до 4М байт при организации «быстрого» обмена с внешними устройствами как с ячейками памяти.

Эти схемы не требуют экотических деталей, доступны для повторения радиолюбителям средней квалификации и позволяют создать мощное программное обеспечение. Прошу откликнуться энтузиастов-программистов, готовых принять участие в разработке операционной системы для «БК+». Схемы расширений могут быть опубликованы.

А. М. КОНЮШЕНКО.
220030 Минск-30, а/я 1 (UC2AAU).

Для эффективного использования БК-0011 необходимо немедленно, пока не накоплено практически никакого программного обеспечения к реализованной на нем операционной системе, создать ОС в стандарте MSX с учетом прогрессивных идей, обеспечивающих легкость расширения и однородность известной MS DOS для IBM PC.

Г. В. МИРОШНИЧЕНКО, В. В. ГОРБУНОВ.
Ленинград.

Имею более 500 программ для БК-0010, хочу обмениваться. Ищу программу вывода информации на телетайп в коде МТК-2.

С. П. ТАЛАНОВ.
612220 Кировская обл., Яранск,
ул. Некрасова, 39, кв. 1.

Обменяюсь программами для БК. Имею систему «Рига», Е-практикум, игровые, обучающе-контролирующие программы. Запись на дискеты (ДВК-3).

А. А. ДЕРБЕНЕВ.
457131 Челябинская обл.,
Троицкий р-н, с. Песчаное, ул. Труда, 7.

Поклонник MSX-Бейсика БК-0010 предлагает обмен программами. Располагаю также широким выбором программ на Фокале, Форте, Т-языке, К-Фокале, в машинных кодах.

Е. Г. ОБЛОЖКОВ.
634062 Томск, пр. Фрунзе, 98, кв. 119.

РЕДАКЦИОННЫЕ СООБЩЕНИЯ

От А. А. Барсукова получено сообщение, что он заканчивает разработку планшета-дигитайзера для ввода графической информации в БК. Основой планшета является визир, индуктивно связанный с полем считывания информации; размер последнего — 250×250 мм, точность считывания положения визира — 0,1 мм, вес прибора 1,5 кг.

Мы надеемся в скором времени опубликовать подробное описание планшета.

В журнале «Радио» (№ 6 за 1990 г.) опубликованы схемы подключения ПЭВМ к телевизорам 2УСЦТ и 3УСЦТ.

М. СЕМИОНЕНКОВ
НИВЦ АН СССР, г. Пущино, Московская обл.

Обучение школьников программированию

Я хотел бы рассказать об опыте обучения школьников программированию, приобретенному за 3 года работы с группой производственного обучения.

Отправная точка

Формирование группы: в группу входил почти полностью один из IX классов пущинской экспериментальной школы. По существу, она была сформирована из желающих, без конкурса. Часть группы уже имела некоторый опыт программирования, приобретенный в кружке пущинского энтузиаста и пионера обучения детей программированию Н. Л. Луниной.

Состав группы: тринадцать юношей и одна девушка.

Количество занятий: 6 ч в неделю.

Технические средства: класс MSX-1 (пятнадцать ученических и одна учительская ЭВМ «Ямаха») и класс СМ-4 с семью дисплеями.

Преподаватель: инженер-программист с четырехлетним стажем работы в научно-исследовательском вычислительном центре и таким же опытом кружковой работы с детьми (не по программированию).

Выбор ЭВМ, языков программирования и среды подготовки программ

Мы использовали в работе оба упомянутых учебных класса и программировали на двух языках. Такой выбор представляется мне целесообразным как для ориентирующихся на профессиональное программирование учащихся, так и для остальной (большей) части группы. Первым полезно осознать общность процедурных языков программирования и проблемы перехода с одной ЭВМ на другую. Для вторых необходимо разнообразие в длительном трехгодичном курсе.

Конкретный выбор языков и среды подготовки программ: Бейсик как основной язык на «Ямахах» и Фортран IV под операционной системой RSX11M (ОС РВ) с редактором текстов «Микромир» (МИМ) как среда, традиционно использовавшаяся в Пущине для обучения школьников, студентов, аспирантов.

Первый год был посвящен программированию на Бейсике, поскольку с него легче начинать. Бейсик достаточно прост для освоения, обладает графическими и рядом других привлекательных возможностей, что позволяет закрепить интерес к компьютеру. Кроме того, очень важно, особенно в начале обучения, предоставить дисплей каждому ученику.

Второй и третий годы были посвящены программированию на Фортране. Два года были выделены для Фортрана с тем, чтобы на последнем году использовать школьников в реальной работе под руководством научных сотрудников. Эта задумка, однако, не удалась: подтвердился опыт других групп, в которых реально могли сотрудничать с научными сотрудниками не более 1—2 человек. В связи с этим разделение времени между языками вызывает задним числом сомнения, поскольку для чисто учебных целей надежная техника с большим числом дисплеев предпочтительнее.

Несколько слов об использовании Бейсика в учебном процессе. Некоторые авторитеты в области программирования и обучения программированию (например, Дейкстра) достаточно категоричны в отрицательной оценке этого языка для обучения. Мотивировка: формирование плохого стиля программирования, который трудно поддается исправлению. Сознаюсь, априори и я был скептически настроен к Бейсику. Однако опыт нашей группы не дал повода для грустных выводов. Да, часть группы писала на Бейсике «лапшой» и несколько человек перенесли этот стиль на Фортран. Но это «факт из нашей биографии», и не более того. Большая часть группы в той или иной степени овладела основами структурного и модульного программирования. Для постановки хорошего стиля программирования важно стремиться воспитать дисциплину программирования у учеников, т. е. добиться понимания того, что программировать надо регулярными методами. Такое воспитание можно проводить независимо от языка программирования. А ущербность Бейсика по меркам современной технологии программирования очевидна. Важные для обучения достоинства Бейсика (см. выше), с моей точки зрения, существенно перевешивают этот недостаток.

Изучение языков программирования и среды подготовки программ

При изучении первого языка программирования проще всего следовать продуманному, «обкатанному» на детях учебнику. Таким учебником для MSX-Бейсика является, на мой взгляд, [1]. К сожалению, в тот момент, когда мы изучали Бейсик, учебник находился еще в работе и мы не смогли в полной мере опереться на опыт автора.

Что касается второго языка, то его изучение не представляет проблемы — с основами Фортрана ознакомились за два четырехчасовых занятия. На первом учащиеся самостоятельно изучали и обсуждали заранее заготовленные короткие программы на Фортране, содержащие все основные элементы языка. Все, что имеет прямые аналоги в Бейсике, было без труда распознано. Незнакомые элементы мне пришлось пояснить. Барьер нового был сломлен. На втором занятии мне осталось лишь рассказать то, «что все уже знают». В процессе последующих занятий многие детали языка разъяснялись ученикам индивидуально. Это удобно и эффективно в аудитории, значительную часть времени предоставленной самой себе. В результате удавалось не перегружать одних и максимально оснащать других.

Схожим образом мы знакомились и с операционной системой. Система была спрятана командным файлом, с помощью которого выполнялись компиляция, построение и запуск задачи. Общее объяснение гласило: «Наберите то-то, и ваша задача выполнится». Дальше каждый продвигался по мере своей любознательности и сообразительности.

О редакторе текстов. По нему мы провели небольшое вводное занятие, после чего все имели возможность совершенствоваться по руководству [2], написанному специально для нашего учебного класса.

Таким образом, по всем вопросам, касавшимся инструментария, мы искали необходимый минимум, предоставляя желающим возможность совершенствоваться.

О подборе задач и дополнительного системного обеспечения

В самой общей форме принципы подбора материала, которыми я руководствовался, достаточно очевидны: поддерживать устойчивый интерес всех учащихся, постепенно усложнять задачи, расширять используемые учащимися системные средства. Не очевидно, однако, как именно этим принципам следовать. Ключ к проблеме — в индивидуальной, интересной и посильной задаче для каждого учащегося. А дополнительные системные средства — это те и только те, которые позволяют решать интересующие его задачи.

Задачи выбирались двумя способами: ребята либо сами их предлагали, либо выбирали из числа предложенных. В первом случае мы совместно с учащимися обсуждали такую постановку задачи, чтобы будущая программа была наиболее интересна, решала задачу максимально широко, имела приемлемый интерфейс с пользователем и т. п. Свои задачи предлагали до половины учащихся.

Естественно, следует стремиться к тому, чтобы как можно большее число учащихся самостоятельно выбирали задачи. Каковы же мотивы самостоятельного выбора? Доминировали два мотива выбора задачи: стремление повторить ранее увиденную программу (обычно игровую или учебную); желание создать инструмент для любимого дела.

Выводы:

учащихся нужно систематически знакомить с ученической и профессиональной программной продукцией;

преподавателю необходимо давать конструктивные советы о возможности использования ЭВМ в сфере интересов ученика.

Проблема индивидуальной задачи быстро стала очевидной на первом году обучения,

поскольку расслоение группы было «запрограммировано» предварительными кружковыми занятиями некоторых учеников. На втором году дисплеев стало меньше, чем учеников, и появилось желание дать каждой паре учеников задачу и дисплей. Однако скоро стало ясно, что в большинстве пар один из партнеров превращается в статиста, который лишь «знает», что происходит, но не приобретает каких-либо навыков. Пришлось вернуться к индивидуальным задачам. При переходе от принципа «1 дисплей — 1 задача» к принципу «1 программист — 1 задача» мы потеряли человеко-часы, проведенные за дисплеем, но приобрели конкуренцию за дисплей со стороны менее активных. Это, конечно, важнее.

Задачи

По нашему опыту, для большинства учащихся (включая тех, кто предлагал свои) приемлемы задачи, в которых пользователь работает с объектами, отображаемыми на экране графического или алфавитно-цифрового дисплея. Примеры: перемещение объекта на плоскости с определенной целью (выход из лабиринта, пересечение минного поля и т. п.), «Ханойская башня», «Морской бой», «Крестики-нолики», разнообразные виды программ для рисования на графическом или алфавитно-цифровом дисплеях, в частности исполнитель Чертежник [3].

В большей степени «на любителя» ориентированы родственные им по форме, но более абстрактные по содержанию программы, реализующие стековый калькулятор [3], игру «Быки и коровы» [4], головоломку «15» и аналогичные ей [5], машину Тьюринга.

Известным спросом пользовались учебные программы: рисование графиков функций, подготовка таблиц функций, тренажер по темам «Полет тела, брошенного под углом к горизонту», «Движение в системе Солнце—Земля—Луна».

Не популярны были программы сортировки и программы реализации математических методов.

Часть учащихся занимались разработкой программ, соответствующих их естественнонаучным склонностям (анализ нуклеотидных последовательностей, расчет электрических цепей).

С задачами согласован набор вспомогательных системных средств. Для работы на СМ-4 школьники изучили комплекс графических подпрограмм ГРАН [6] для работы с графическим дисплеем и пакет CRT для работы с экраном алфавитно-цифрового дисплея. Для работы на «Ямахах» достаточно средств Бейсика.

О подборе совокупности задач. Подбирая задачи каждому учащемуся, я стремился к возможно более полному покрытию следующего списка понятий и навыков:

работа с одномерными и двумерными массивами;

моделирование более сложных структур данных (стеки и др.);

создание модуля (набора подпрограмм, работающего с определенной структурой данных);

работа с клавиатурой;

работа с графическим экраном;

работа с алфавитно-цифровым экраном;

работа с датчиком случайных чисел.

Самообучение

За время занятий мне довелось наблюдать интересный феномен самообучения, не характерный для традиционного обучения в школе. Механизм очень прост: учащиеся общаются на занятиях между собой, обмениваются опытом. Более подготовленные консультируют менее подготовленных. Многие проблемы моих подопечных решались таким образом «сами собой», без моего участия. Создалась комфортная для учащихся обстановка, когда вопрос не обязательно адресовать преподавателю, а можно сначала обсудить с товарищем. Консультантов такая деятельность возвышала в собственных глазах, укрепляла их авторитет в коллективе, и они без особых сожалений шли на некоторые жертвы своим временем. Консультанты иногда брались «донести» до преподавателя вопрос, возникший в аудитории, если не могли разобраться с ним самостоятельно. Мой КПД при этом возрастал, поскольку я не ограничивался чисто прагматическим частным рецептом, а объяснял вопрос по возможности глубоко. Последующие аналогичные вопросы после такого объяснения обычно до меня не доходили.

Простой и очень эффективный способ самообучения не требует каких-либо затрат со стороны преподавателя, требуется лишь не препятствовать свободному общению учащихся (естественно, до тех пор, пока это не мешает окружающим). Самообучение — существенный резерв разнородной группы.

Экзамен

Основу экзамена составляла защита программы, которая выбиралась ребятами по своему усмотрению из числа отлаженных за последние два года обучения. Очевидно, что распечатка достаточно большой отлаженной программы говорит о навыках программирования больше, чем умение кодировать корот-

кие примеры в экстремальных условиях экзамена. До начала защиты каждый учащийся проходил небольшое собеседование, на котором он рассказывал о своих успехах в программировании за время обучения. Имевшие большой интересный «послужной список» получили высшую оценку без экзамена (выпускники, получившие «автомат», награждены на выпускном вечере Почетной грамотой и годовой подпиской на журнал «Квант»). Были на экзамене и билеты, содержавшие два вопроса по Фортрану, но они имели второстепенное значение для оценки. Это был скорее повод несколько систематизировать знания ученика. В случае, когда защищаемая программа содержала разнообразные средства языка, опрос по билету просто отпадал за ненадобностью.

Выводы

Начну с главного (на мой взгляд) вопроса: не было ли это производственное обучение потерей времени для детей и преподавателя? Убежден, что нет, и вот почему.

1. Такое производственное обучение было интересно большинству учащихся (а не секрет, что в школе учиться довольно скучно) и определенно расширяло их кругозор.

2. Для части учащихся наши занятия имели большое значение в плане профориентации. Они смогли не только достичь некоторого уровня программирования (в отдельных случаях очень высокого даже по студенческим меркам), но и получить представление о программировании как профессиональной деятельности, в которой есть не только забавное, красочное, интересное. Это безусловно поможет им в более сознательном выборе профессии.

Те учащиеся, которые не рассматривали программирование как возможную профессиональную деятельность, в случае поступления в вузы столкнутся с программированием, будучи уже подготовленными.

3. Наконец, для подростков, как мне кажется, полезно общение с профессионалами, независимо от профессии последних. Это некоторый прорыв во взрослый мир, и он имеет определенное воспитательное значение.

Что касается «арифметических» итогов, то из группы в 14 человек, сформировавшейся на первом году обучения, 13 прошли полный курс. Из этих 13 один учащийся воспользовался своим правом не сдавать экзамен и удовлетворился годовой оценкой. Из 12 явившихся на экзамен 4 получили «автомат» за программы длиной свыше 500 операторов или большое число программ длиной 100—300 операторов. Остальные защищали программы длиной в основном 100—200, реже — 300—400 операторов. Обычно это была единственная из сколько-нибудь серьезных программ ученика. Группа произвела хорошее впечатление на экзаменационную комиссию, и десяти выпускникам была поставлена оценка «отлично».

И наконец, удовлетворен ли я своей деятельностью, соотношением затрат и результатов? Не вполне. Не будем задерживаться на внешних по отношению к группе обстоятельствах: отсутствии необходимого опыта преподавателя, недостатке времени на высококачественную подготовку занятий, неудовлетворительной работе техники. Остановимся на другом: группа, как уже говорилось, не была сформирована на основе широкого конкурса. Последствия, думаю, очевидны, хотя группу в целом отнюдь не назовешь слабой или равнодушной к предмету.

Для себя одним из главных результатов я считаю осознание возможности подготовки группы старшеклассников на высоком, приближающемся к профессиональному, уровне. Такая группа могла бы отбираться на конкурсной основе из учащихся, скажем, VIII—IX классов всех пущинских школ. Размер группы может быть поменьше. Поскольку занятия для учеников разных классов и школ необходимо проводить во внеучебное время, можно несколько сократить время занятий. Условия для подбора и обучения такой группы в Пущине вполне подходящие: есть техника, кружки программирования, квалифицированные программисты, накопившие опыт преподавания. У многих детей есть возможность познакомиться с компьютером с помощью родителей и знакомых — научных сотрудников и инженеров.

Предоставление способным детям возможности для развития своих способностей уже в школьные годы вполне отвечает современным тенденциям в образовании, и автор надеется принять посильное участие в этом деле.

Литература

1. *Лунина Н. Л.* Алгоритмический язык Бейсик для ЭВМ «Ямаха». Пущино, 1987.

2. *Седов В. Ю.* Редактор текста «Микромир-85». Пущино, 1986.

3. *Лунина Н. Л.* Обучение школьников программированию. Пущино, 1986.

4. *Гик Е. Я.* Занимательные математические игры. М.: Знание, 1987.

5. Квант. 1987. № 1—12.

6. Каталог программных средств. Фонд алгоритмов и программ НИВЦ АН СССР. Пущино, 1987.

Л. ВОРОНОВА,
канд. техн. наук,
Владимирский политехнический институт

Преподавание алгоритмических языков

В настоящее время специалисты многих профессий озабочены отсутствием методик обучения, позволяющих в короткий срок овладеть использованием алгоритмических языков в своей области. Традиционная технология и общедоступная литература для этого не годятся, так как изначально ориентированы не на «широкого читателя», а на подготовку специалистов в области математических методов и ВТ (ее создания и эксплуатации). Поэтому становятся целесообразными специальные исследования по разработке методов обучения алгоритмическим языкам, аналогичные проводящимся много лет в отношении естественных языков.

В статье описываются некоторые приемы и указываются методические материалы, разработанные частично в соавторстве с Р. И. Дубовым [1], применяющиеся на кафедре прикладной математики института, которые позволяют сократить время обучения алгоритмическим языкам без ущерба для качества.

Их можно рекомендовать для самообучения языку, преподавателям курсов повышения квалификации, аспирантам различных специальностей, использовать в обучении студентов технических вузов и техникумов особенно на тех специальностях, где программирование преподается по сжатой программе.

Предлагаемые материалы могут быть интересны учителям по ОИВТ школ и училищ, тем более что и сами учителя порой нуждаются в «быстрой» методике повышения собственной квалификации в этой области. Наряду с совершенствованием навыков алгоритмического мышления учителю необходима «быстрая» схема знакомства с любым выбранным для обучения алгоритмическим языком, чтобы немедленно суметь применить его практически хотя бы для простейших учебных задач (с последующим более глубоким его освоением).

Приведенные приемы «хорошо работают» на алгоритмических языках, ориентированных на задачи вычислительного характера (Бейсик, Фортран, PL, Паскаль и др.). Контрольные примеры и иллюстрации даются на материале по Фортрану.

Итак, чем располагает преподаватель, обучая слушателей программированию? Распространенными книгами — руководствами по алгоритмическим языкам, выдержанными в стиле традиционной методики: описание ведется хотя бы отчасти с классификацией по конструкциям языка. Например, в руководствах по Фортрану под заголовками «Оператор IF», «Оператор DO» и т. д. описывается, что означают эти операторы и как, в каких ситуациях их можно применить для решения различных задач. Это принцип словарей, предназначенных для перевода текстов с незнакомого языка на родной. В них для каждого иностранного слова приводятся его возможные переводы (возможные применения). Такие руководства и словари очень полезны для чтения и разбора готовых программ, так как они полнее описывают конструкции алгоритмического языка.

Вначале в распоряжении обучаемого в лучшем случае имеется готовый алгоритм решения своей задачи, составленной на русском языке или в графической форме, либо просто минимально возможный набор элементарных алгоритмических предписаний, различные комбинации которых будут давать алгоритмы решения. И практически отсутствие знаний нужного алгоритмического языка! С чего начать? С абстрактного изучения языка без привязки к своей работе или с первоочередного знакомства с его возможностями применительно к данной работе? Второе более привлекательно, так как явно связано с целями обучения, но практически нереализуемо для новичков в программировании при наличии только учебно-методической литературы указанного выше типа.

В подобной ситуации для ускорения перевода текста со знакомого языка (например, с русского) на менее знакомый целесообразно использовать словари с обратным порядком приводимых слов и объяснений (назовем их инверсными). В них каждому русскому слову соответствуют его возможные переводы на требуемый иностранный язык. С помощью таких словарей можно активно использовать иностранный язык на более ранней стадии изучения (при условии освоения его алфавита и грамматики).

Р. И. Дубовым был предложен аналогичный принцип составления инверсных словарей — руководств для перевода общепонятных записей операций алгоритма на алгоритмические языки для ЭВМ. Совместные исследования по уточнению этого принципа, разработке конкретного руководства для преподавания алгоритмических языков, апро-

бация в учебном процессе, дальнейшее развитие этих идей показали, что данный подход целесообразно использовать для быстрого освоения языка на начальном этапе обучения.

Переход от прямого руководства-справочника к инверсному является не просто процедурой перестановки. Как известно, при составлении обычных словарей для разговорных языков нельзя ограничиться такой процедурой из-за несовпадения границ понятий в сопоставляемых языках. Аналогичные трудности проявляются и в алгоритмических языках и требуют специального изучения. Даже формулировка типичных ситуаций, используемых при построении различных алгоритмов (именно по их классификации строится инверсное руководство), названия операций, подлежащих переводу на алгоритмический язык, требуют серьезной работы с целью выражения этих понятий в удобной для использования форме. Эти трудности являются, вероятно, причиной того, что ранее руководства по алгоритмическим языкам, полностью выдержанные в инверсном стиле, не были созданы. На нашей кафедре составлен вариант инверсного руководства по обучению Фортрану в пределах программы технических вузов по специальностям, требующим знаний основ программирования на Фортране [2].

В какой момент процесса начального обучения можно предложить использовать подобное руководство? В нем не рассматриваются понятие алгоритма, основные приемы и способы алгоритмизации решения задач и типовые структуры алгоритмов для различных вычислений. Предполагается, что обучаемые получают предварительно (на соответствующих лекциях и семинарах) необходимые сведения по теории алгоритмизации, и перед ними встает вторая задача: перевести последовательно все операции алгоритма на язык, пригодный для ввода его в ЭВМ и автоматического исполнения. Параллельно с этим надо уметь представлять данные на входе задачи и заказывать форму выходных данных в соответствии с правилами выбранного языка. Вот на этом этапе и предлагается соответствующим образом подавать лекционный материал по программированию, а также самостоятельно пользоваться студентам и учащимся подобными справочниками, отыскивая нужные операции по мере необходимости. Одновременно происходит и запоминание, усвоение соответствующих Фортран-конструкций, т. е. процесс самообучения программированию. Материал классифицируется строго по «задачам» (по элементарным предписаниям), из которых складываются алгоритмы, записанные на обычном языке, и даются указания по их выражению на Фортране. Приводятся дополнительные возможности ускорения и улучшения процесса обучения алгоритмическим языкам. При наличии такого руководства обучение программированию на начальном этапе сводится к изучению этого руководства в применении к алгоритму решения конкретной задачи, составленному на естественном языке.

Опыт работы в области алгоритмизации решения самых различных вычислительных задач показывает, что все многообразие вычислительных схем может быть сведено к комбинациям ограниченного набора типовых алгоритмических структур. А последние, в свою очередь, могут быть составлены из конечного набора базовых элементарных предписаний (действий) алгоритмов над конечным набором типовых элементарных конструкций.

Назовем возможный набор элементарных предписаний: ввод данных, вычислительная операция, операция сравнения (выбор пути последующих вычислений), вывод результатов, останов. Указанные операции могут быть выполнены над такими элементарными конструкциями, как числа и переменные различных типов.

Поэтому, если подходить к изучению алгоритмического языка с практической позиции, требующей максимально быстрого перехода от алгоритма к программе, пользователь в первую очередь будет интересоваться возможностями языка по представлению именно этого набора элементарных конструкций и предписаний. Именно для него важно в первую очередь получить минимально необходимые знания по «переводу» общепонятных записей алгоритма на выбранный алгоритмический язык. Содержание руководства отвечает этому требованию: рассматривается общее оформление алгоритма в виде самостоятельной программной единицы на бланке; затем — элементарные конструкции, входящие в операции (числа, переменные), варианты их использования в алгоритмах и соответствующие «переводы» на Фортран; затем приводятся собственно операции (базовые) с возможными модификациями и варианты «перевода» на алгоритмический язык. Во второй части руководства рассматриваются более сложные алгоритмические действия и возможности их представления на Фортране.

Основные конструкции и операции, подробно приведенные по тексту руководства-справочника [2], представлены соответственно своими фрагментами, каждый из которых детализируется по четырем позициям.

Первая — запись элементарной конструк-

ции или операции алгоритма в обобщенном виде на русском языке.

Вторая — «перевод» этой записи на Фортран (для уменьшения количества ошибок здесь же даются пояснения на русском языке, взятые в фигурные скобки).

Третья — типовой демонстрационный пример в обычной записи: на математическом языке с русским сопроводительным текстом (примеры подобраны на основании опыта преподавания так, чтобы пользователь мог по ним получить ответы на возможно большее число своих вопросов).

Четвертая — программирование этого примера на Фортране.

Особенности использования какой-либо конструкции Фортрана указываются в сносках и примечаниях для соответствующего фрагмента справочника.

Приведем конкретный фрагмент справочника для такой элементарной конструкции, как любое смешанное число, представленное в формѐ с фиксированной запятой (аналогичным образом представляются и фрагменты по переводу основных операций в соответствующие операторы Фортрана).

Смешанное число с фиксированной запятой

1. **Общее название числовой конструкции в обычной математической терминологии.** Действительное, смешанное (дробное) число с фиксированной запятой.

2. **Общее название той же числовой конструкции в Фортране.** Вещественное (реальное) число с фиксированной десятичной точкой — как обычная математическая запись дробного числа при замене десятичной запятой на десятичную точку (знак + и нулевая дробная или целая часть могут опускаться); максимальное количество значащих цифр числа зависит от используемой ЭВМ*.

3. **Примеры записи чисел в математической форме:**
   1) +0,325; 2) —1,41; 3) —2,00.

4. **Примеры записи тех же чисел в Фортране:**
   1) +0.325 | 0.325 | .325 2) —1.41 3) —2.00 | —2.

Однако многолетний личный опыт преподавания подобных дисциплин говорит о том, что на начальном этапе обучения даже такие руководства для запоминания являются избыточными, затеняющими основную канву перехода от алгоритма к программе. Поэтому при обучении весьма целесообразно рекомендовать слушателям составлять (по лекциям и руководству) специальную «шпаргалку-подсказку», содержащую сведения первой необходимости из алгоритмического языка. Автор рекомендует следующий ее вид: это форматный лист полуватмана (в размер развернутого тетрадного листа), разграфленный на четыре столбца (см. фрагмент листа на рисунке). В первом столбце перечисляются на русском языке все пять названных выше предписаний алгоритмов (конкретными простейшими примерами). Во втором столбце — соответствующие их графические представления (если кроме словесной формы записи алгоритма надо уметь представлять его блок-схему). В третьем — запись каждой операции на алгоритмическом языке (в частности, пример приведен на Фортране). В четвертом — краткие сведения об основных конструкциях, используемых в демонстри-

*Числа в форме вещественных констант с фиксированной запятой (в так называемой форме F) представляются в машине приближенно. Для ЭВМ серии ЕС максимальное количество значащих цифр числа стандартной длины равно 7.

Основные операции алгоритмов	Графическое представление (для блок-схем)	Запись операций на Фортране (операторы)	Дополнительные сведения
1. Ввод данных (например, ввести смешанное число 0.5 и целое 10 для переменных *a* и *k*)	. . . . (после выполнения ввода считается: *a*=0.5, *k*=10)	12345678... ... m READ(n,m)A,K m FORMAT(F3.1,2X,I2) ... {исходные данные после текста программы 0.5__10 n - номер канала ввода (используемое внешнее устройство); m - метка оператора, содержащего формы записи вводимых данных - формата (см. след. столбец)}	*a*=0.5== 0.5 вещественное с фикс. точкой. Описание формы: F3.1, где F код, w=3 длина числа в позициях (включая точку), d=1 длина дробной в позициях; *k*=10== 10 - целое. Описание формы: I2, где I код, w=2 длина числа в позициях, 2X - два пробела между числами, где X код, w=2 количество пробелов.
. . .	. . .	. . .	. . .

руемой операции: числах (и подборе их форм — форматов), переменных (и описании их типов), массивах (и использовании их элементов).

Таким образом, на небольшой лист, обозримый одним взглядом, студент или учащийся постепенно самостоятельно выписывает самые основные сведения по алгоритмическому языку в инверсном варианте, которые оказываются вполне достаточными для быстрых напоминаний при написании простейших программ, для запоминания, а также весьма полезны для более глубокого знакомства с языком, когда рекомендуется пользоваться уже подробным инверсным руководством. А главное, лист-вкладыш можно вставлять при необходимости в любую рабочую тетрадь, книгу, учебник по другой дисциплине, требующей использования ЭВМ. Желательно его иметь и в самом справочнике.

Описанные обучающие материалы не исключают, а дополняют использование обычных руководств. Интересно то, что традиционная учебно-методическая литература, используемая на последующих этапах обучения, воспринимается неспециалистами в области программирования уже без затруднений.

Литература

1. *Дубов Р. И. Воронова Л. М., Когдов Н. М.* и др. Организационные вопросы применения вычислительной техники в учебном процессе // Средства обучения в высшей и средней школе. Вып. 3 М.: НИИ ВШ, 1985.

2. Ускоренное обучение программированию на алгоритмических языках. Фортран, начальные сведения / Под. ред. Л. М. Вороновой. Владимир.: ВПИ. 1987.

С. ФАДЕЕВ
Ленинградский государственный университет,
М. ФАДЕЕВА
Ленинградский санитарно-гигиенический медицинский институт

От диалога с компьютером — к диалогу с больным

Иностранные студенты, обучающиеся в вузах СССР, испытывают известные трудности, связанные с преподаванием на русском языке. Преодолению этих трудностей в значительной мере способствуют занятия на кафедрах русского языка для иностранцев. В настоящее время эти кафедры строят учебный процесс на конкретных материалах по специальностям, изучаемым иностранными учащимися.

При обучении студентов-медиков одной из важных задач является формирование умения вести профессиональный диалог на различные темы. Однако в традиционных условиях преподаватель русского языка не имеет достаточного времени, чтобы довести до автоматизма различные типа речевых действий профессионального диалога у каждого учащегося. Между тем такое положение можно существенно изменить, если преподаватель возьмет себе в помощники компьютер.

Вопреки существующему мнению, что компьютер можно применять лишь для языковой тренировки, т. е. для формирования и автоматизации различных языковых (лексических, грамматических, орфографических и др.) навыков, мы решили использовать его и для формирования диалогических умений. Тем более, что общение студента с компьютером происходит в диалоговом режиме. Это, конечно, диалог в письменной форме, но сейчас уже общеизвестно, что существует явление переноса навыков и умений из одних видов речевой деятельности в другие, т. е. умения и навыки, сформированные в письменной речи, могут быть перенесены и в устное диалогическое общение (см., например, [1]).

При обучении с использованием компьютера в зависимости от степени активности обеих сторон диалога можно различать [2]:

диалог, при котором активным является компьютер;

диалог с переходом активности от компьютера к обучаемому и наоборот;

диалог, при котором активным является обучаемый.

Первый подход реализуется в большинстве существующих компьютерных программ по русскому языку, когда компьютер предъявляет обучаемому определенную дозу информации, проверяет затем ее усвоение с помощью вопросов и заданий и, проанали-

зировав ответы студента, соответствующим образом оценивает их.

Однако для студентов-медиков гораздо больший интерес представляют программы двух последних типов. Существует мнение, что научиться диалогической речи нельзя путем заучивания готовых диалогических текстов, помещенных в учебниках и учебных пособиях [3]. Это утверждение в основе своей верно. Однако следует отметить, что оно вряд ли справедливо при обучении профессиональному стандартизированному диалогу, когда общение строится на одних и тех же (с незначительными вариантами) речевых клише и практически не меняется даже при смене собеседников.

Поясним это на примере фрагмента сценария ситуативно-моделирующей программы «Врач и больной», предназначенной для иностранных студентов II—III курсов лечебных факультетов медицинских вузов СССР. Данная программа направлена на формирование у студентов-иностранцев умений ведения профессионального диалога со стационарным больным и правильного оформления записи в медицинской карте стационарного больного.

На первом этапе программы инициатива ведения диалога принадлежит компьютеру. Сначала он предлагает студенту ознакомиться с образцом диалога, который обычно происходит между врачом и больным при заполнении титульного листа медицинской карты.

Приведем фрагмент диалога:

1.— Ваша фамилия?
— Петров.
2.— Имя, отчество?
— Сергей Павлович.
3.— Возраст?
— 40 лет.
4.— Вы живете в этом городе?
— Да.
5.— Ваш домашний адрес?
— Красногвардейский проспект, 15, квартира 148.
6.— Домашний телефон есть?
— Есть: 218-44-32 и т. д.

Диалог появляется на экране не целиком, а последовательно — реплика за репликой, с небольшими временными интервалами. После ознакомления студента с диалогом-образцом компьютер берет на себя функции врача, т. е. ведущего диалог-расспрос, а обучаемому предлагает роль пациента (формирование диалогических умений реактивной речи);

— Я — врач. Вы — больной. Я буду задавать Вам вопросы, Вы должны давать краткие и четкие ответы.

Таким образом, диалог «прокручивается» еще раз, после чего компьютер предлагает студенту поменяться ролями, т. е. инициатива переходит от компьютера к студенту, который теперь сам формулирует вопросы и задает их компьютеру (формирование диалогических умений инициативной речи). В распоряжении последнего для ответа на тот или иной вопрос «врача» имеются специальные массивы данных, подготовленные разработчиком программы. При ответе на конкретный вопрос компьютер, используя генератор случайных чисел, выбирает из соответствующего массива необходимый ответ, т. е. при многократном повторении диалога каждый раз выдается новая информация. Таким образом компьютер имитирует разное количество больных, пришедших на прием к врачу, и максимально приближает учебную ситуацию к реальной.

Когда из инициативных вопросительных реплик студента-«врача» исчезнут ошибки, т. е. компьютер добьется от студента достаточного автоматизма реплицирования, инициатива ведения диалога снова переходит к компьютеру. На этот раз на экран выводится титульный лист стандартной медицинской карты стационарного больного и компьютер показывает, как ответы больного используются для заполнения медицинской карты.

Затем на экран выводится чистый бланк медицинской карты, а компьютер снова передает инициативу студенту. Теперь задача студента усложняется, так как ему необходимо не только вести диалог с «больным», но и параллельно заполнять медицинскую карту на этого больного.

Компьютер, как уже отмечалось, позволяет повторять эту ситуацию сколько угодно раз, способствуя выработке устойчивых навыков у иностранных студентов-медиков.

Программа «Врач и больной» относится, как уже говорилось выше, к типу ситуативно-моделирующих программ по русскому языку как иностранному. В настоящее время на кафедре русского языка ЛСГМИ ведется интенсивная разработка аналогичных компьютерных программ и по другим темам профессионального медицинского общения, например по теме «Бронхит». В данной программе учащийся должен расспросить больного (компьютер) о симптомах заболевания и поставить соответствующий диагноз. Программа включает в себя целый ряд взаимосвязанных этапов, выделенных нами на основе плана, предложенного В. Дьяковой [4]:

1. Жалобы больного.
2. История настоящего заболевания.
3. История жизни больного.
4. Настоящее состояние больного.
5. Предположительный диагноз.

Разработка подобных программ представляется нам очень перспективным и необходимым делом, так как с их помощью до-

стигается высокая степень индивидуализации обучения иностранных учащихся в процессе формирования у них инициативных и реактивных диалогических умений профессиональной речи, а также существенно сокращается время, необходимое для выработки данных умений при полном охвате всех учащихся группы.

Литература

1. *Сердюков П. И.* Как составить алгоритм упражнений для ЭВМ // Иностранные языки в школе. 1986. № 6. С. 26—31.

2. *Бъчваров С.* Опыт и перспективы Софийского университета по применению компьютеров в учебном процессе // ЭВМ в вузе: Межвуз. сб. научн. трудов. Новосибирск: НГУ, 1984.

3. *Шатилов С. Ф.* Методика обучения немецкому языку в средней школе. М.: Просвещение, 1986.

4. *Дьякова В. Н.* Пособие по развитию речи. Диалог врача с больным. Заполнение медицинской карты стационарного больного. М.: Русский язык, 1986.

Г. ШРАЙБЕР

Предлагаю программы для ПМК

В учебном пособии по ОИВТ под редакцией А. П. Ершова и В. М. Монахова есть «Библиотека алгоритмов». По некоторым из них можно составить программу для ПМК. Рассмотрим одну из них.

Для алгоритма нахождения площади криволинейной трапеции, ограниченной прямыми $x=a$, $x=b$, $y=0$ и кривой $y=f(x)$, при условии, что для всех x из отрезка $[a; b]$ значения $f(x)\geqslant 0$

```
алг ПЛОЩАДЬ (вещ a, b, S, нат n)
    арг a, b, n
    рез S
нач вещ h, x, нат i
    h:=(b-a)/n; S:=0; x:=a; i:=1
    пока i≤n
    нц S:=S+h·f(x); x:=x+h; i:=i+1
    кц
кон
```

программа для МК-61 приведена справа.

Начиная с команды 32, ввести подпрограмму для вычисления $f(x)$, закончив ее командами x→П 2; В/О.

Ввести: 0→RG0; 1→RG1; a→RGa; b→RGb; n→RGc. Затем, нажав клавиши В/О и С/П, запустить программу на счет. Для получения ответа нажать клавиши П→x 0.

Дополнительные сведения: h хранится в RGd; S — RG0; x — RGe; i — RG1; f(x) — RG2.

Могу предложить программы на основе алгоритмов пособия под редакцией А. П. Ершова, В. М. Монахова. *Адрес:* 273000, г. Кодыма, Одесская область, пл. Гусева, д. 15.

Г. И. Шрайбер

Адрес	Команда		Код
00	П→x	b	6L
01	П→x	a	6—
02		—	11
03	П→x	c	6[
04		÷	13
05	x→П	d	4Г
06	П→x	a	6—
07	x→П	e	4E
08	П→x	c	6[
09	П→x	1	61
10		—	11
11	F x≥0		59
12	31		31
13	ПП		53
14	32		32
15	П→x	2	62
16	П→x	d	6Г
17		×	12
18	П→x	0	60
19		+	10
20	x→П	0	40
21	П→x	e	6E
22	П→x	d	6Г
23		+	10
24	x→П	e	4E
25	П→x	1	61
26		1	01
27		+	10
28	x→П	1	41
29		БП	51
30		08	08
31		С/П	50

Э. ГАЛЬВАС, В. ФАСТ

Физические задачи в курсе информатики

В курсе информатики рассматривается построение алгоритмов для некоторых физических задач. Но, на наш взгляд, выбранные задачи не стимулируют поиск общих подходов к их решению. Задачи должны подбираться таким образом, чтобы разработка алгоритмов помогла учащимся увидеть и усвоить внутреннюю логику решения задач определенных типов.

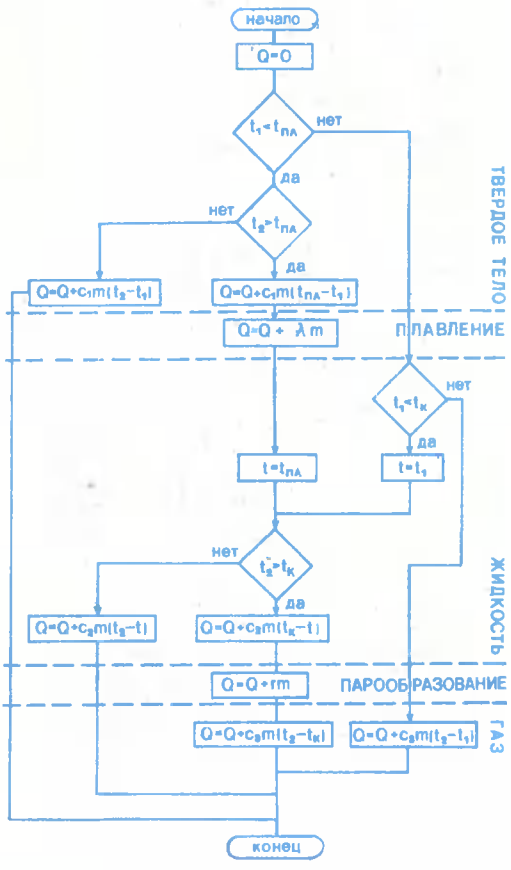
В этом смысле очень интересной представляется нам задача из журнала «Физика в школе» (1984, № 3).

Задача. Тело из известного вещества массой m находится в состоянии с температурой t_1. Какое количество теплоты потребуется для доведения этого тела до состояния с температурой t_2 ($t_2 > t_1$)?

Рассмотрим три случая.

1. Вещество не переходит в другое агрегатное состояние.
2. Агрегатное состояние меняется один раз: а) твердое→жидкое или б) жидкое→газообразное.
3. Агрегатное состояние меняется два раза: твердое→жидкое→газообразное.

На рисунке представлена блок-схема, которая учитывает каждый из них. Таким образом, можно разработать алгоритм, применимый к решению целой группы задач на составление уравнения теплового баланса.



Предлагаем вниманию читателей программу и вариант решения.

```
    READLN (C2);
  UNTIL C2 > 0;
  REPEAT
    WRITE ('     C gas  : ');
    READLN (C3);
  UNTIL C3 > 0;
  REPEAT
    WRITE ('     q plav : ');
    READLN (L);
  UNTIL L > 0;
  REPEAT
    WRITE ('     q par  : ');
    READLN (R);
  UNTIL R > 0;
  WRITELN ('NAGREVANIE TELA:');
  WRITE ('     OT TEMPERATURY:');
  READLN (T1);
  REPEAT
    WRITE ('     DO TEMPERATURY:');
    READLN (T2);
  UNTIL T2 > T1;
  REPEAT
    WRITE ('JEDINICA KOLICHESTVA TEPLOTY RES: 1 - Dsh,
                                              2 - Kal, 3 - Kkal ');
    READLN (KR);
  UNTIL (KKT > 0) AND (KKT <= 3);
END;

FUNCTION TVERD(TT:REAL):REAL;
BEGIN
  TVERD:=C1*M*(TT-T1)
END;
FUNCTION SHIDK(TT,TTT:REAL):REAL;
BEGIN
  SHIDK:=C2*M*(TT-TTT)
END;
FUNCTION GAS(TT:REAL):REAL;
BEGIN
  GAS:=C3*M*(T2-TT)
END;
FUNCTION PLAV:REAL;
BEGIN
  PLAV:=L*M
END;
FUNCTION PAR:REAL;
BEGIN
  PAR:=R*M
END;

BEGIN
  VVOD;                    {VVOD ISQODNYQ DANNYQ}
```

```
WRITELN ('    IDET RASSCHET...');

IF T2<TPL THEN                  Q:=TVERD(T2)
ELSE
   IF T2<TKIP THEN
      IF T1>TPL THEN
                                Q:=SHIDK(T2,T1)
      ELSE
                                Q:=TVERD(TPL)+PLAV+SHIDK(T2,T1)

   ELSE
      IF T1>TKIP THEN           Q:=GAS(T1)
      ELSE
         IF T1>TPL THEN         Q:=SHIDK(TKIP,T1)+PAR+GAS(TKIP)
         ELSE
                                Q:=TVERD(TPL)+PLAV+
                                   SHIDK(TKIP,TPL)+PAR+GAS(TKIP);

IF KKT=KR THEN                  (PEREVOD RESULTATA)
ELSE
   IF KKT>KR THEN
      BEGIN
      IF KR=1 THEN Q:=Q*4.19;
      IF KKT=3 THEN Q:=Q*1000
      END
   ELSE
      BEGIN
      IF KR=3 THEN Q:=Q/1000;
      IF KKT=1 THEN Q:=Q/4.19
      END;

   IF KR=1 THEN EDR:='Dsh'
          ELSE
          IF KR=2 THEN EDR:='Kal'
                  ELSE EDR:='Kkal';
   WRITELN ('  TREBUEMOE KOLICHESTVO TEPLOTY: ',Q:9:3,' ',EDR);
END.
```

```
Running
VESCHESTVO TELA: VODA
SISTEMA JEDINIC:
   JEDINICA MASSY: 1 - Gramm, 2 - Kgm, 3 - Tonna: 2
   JEDINICA TEMPERATURY: 1 - Gradus C ili K,
                         2 - Gradus Farengate 1
   JEDINICA KOLICHESTWA TEPLOTY: 1 - Dsh, 2 - Kal, 3 - Kkal 3
KARAKTERISTIKI TELA:
     Massa  : 2
     T plav : 0
     T kip  : 100
     C tverd: .5
     C shidk: 1
     C gas  : .3
     q plav : 80
     q par  : 539
NAGREVANIE TELA:
     OT TEMPERATURY:-10
     DO TEMPERATURY:100
JEDINICA KOLICHESTVA TEPLOTY RES: 1 - Dsh, 2 - Kal, 3 - Kkal 3
    IDET RASSCHET...
  TREBUEMOE KOLICHESTVO TEPLOTY:  1448.000 Kkal
```

В. МАНЗЮРА

Компьютер и ударение

Драматическое повествование в одном действии с эпилогом

Действующие лица:

Лектор, специалист по культуре речи.
Сидоров, кандидат в депутаты.
Мария Ивановна, учительница.
Алеша, ученик VII класса.

Лектор. Тема нашей беседы — «Компьютер и ударение». Все вы, конечно, понимаете, что грамотная русская устная речь включает в себя владение нормами ударения. Особенно это важно для публичной речи. Бывают случаи, когда ошибки в ударении снижают авторитет оратора.

Сидоров. Это точно. Я вот в одном выступлении сказал «мы́шление», а в зале зашикали, что это, мол, устаревшее ударение, надо говорить «мышле́ние», и слушать меня уже стали не так внимательно. Но откуда мне знать правильное ударение, если в школе этому не учат?

М. И. Неправда. В школьных учебниках есть списки слов, озаглавленные «Произноси правильно!». Кроме того, я да и другие учителя вывешиваем в кабинете списки слов с правильным ударением и периодически их обновляем. Спросите у Алексея.

Алеша. Да, это так. Но заучивать эти списки очень трудно. Почему, например, надо говорить «цеме́нт», «докуме́нт», хотя в словах «парла́мент», «регла́мент» ударение на предпоследнем слоге? Если бы для таких трудных слов были правила...

М. И. Я согласна с Алешей. Ведь мне самой часто приходится определять правильное ударение в словах по словарям. А в словарях нередко такой разнобой! Смотрела я как-то слово «петля» (по телевизору шел фильм с таким названием, и дети спрашивали, как его правильно произносить). В одном словаре указано: петля́[1]. В другом даны варианты: петля́ и пе́тля, но петля́ — на первом месте, как предпочтительное[2]. Третий словарь приводит варианты наоборот: на первом месте — пе́тля[3]. А ведь все три словаря вышли почти в одно время.

Сидоров. Раз уж словари между собой договориться не могут да еще и признают для некоторых слов варианты ударения, то не проще ли признать правильными любые варианты для всех слов? Пусть кто хочет, говорит «мы́шление», а кому это не нравится — «мышле́ние». И потом, раз все так неоднозначно и запутанно, о каком компьютерном моделировании может идти речь?

Лектор. Прекрасно, друзья мои, что у вас возникло столько вопросов. А ответ свой я начну с того, что разнобой в ударении допускать нельзя, как, кстати, и в других сферах языка, иначе мы будем, вместо того чтобы воспринимать содержание речи, дивиться ее звучанию...

Кроме того, нормы ударения не придумываются учеными, а отражают законы и правила самого языка. Я согласен: правила эти изучены еще недостаточно, и отсюда частично вытекает разноречивость в рекомендациях словарей. Но многое все же сделано. Закономерности русского ударения наиболее полно и вместе с тем компактно описаны известным лингвистом А. А. Зализняком[4]. Однако сразу скажу, что правила ударения очень сложны и содержат немало исключений.

Алеша. Но чем объяснить эти исключения в самом языке?

Лектор. Объясняются они тем, что система русского ударения в последние несколько

веков перестраивается на новый лад, и все исключения и колебания как раз и отражают этот процесс.

С и д о р о в. Затянулась что-то перестройка... И все-таки, уважаемый до́цент...

М. И. Доце́нт.

Л е к т о р. Я профессор.

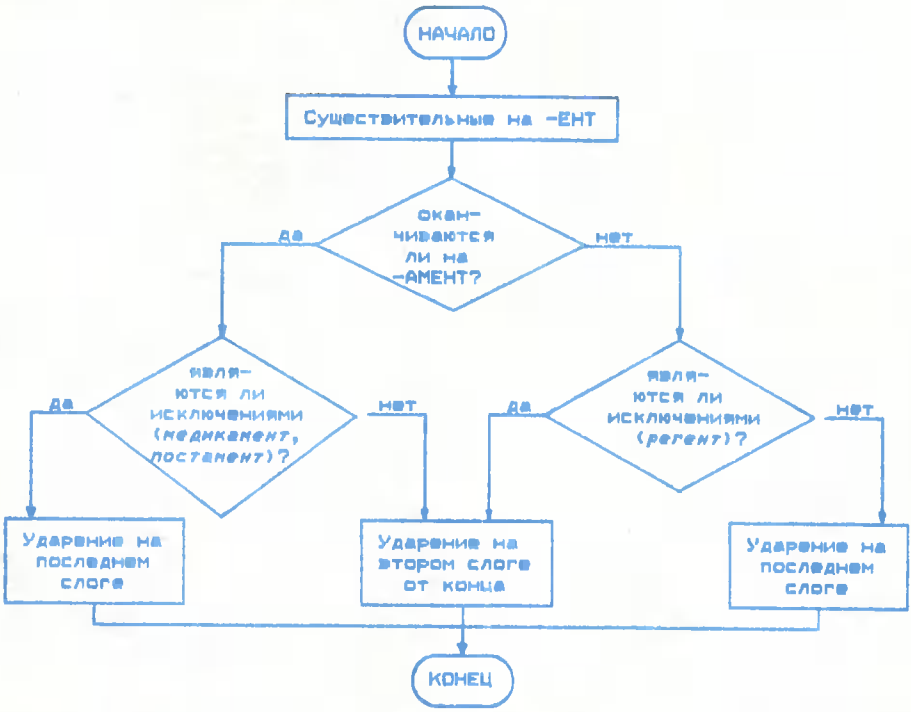
С и д о р о в. Тем более. Я так и не пойму, как всю ту неопределенность, я бы даже сказал — нестабильность, о которой вы говорите, можно изобразить на компьютере.

Л е к т о р. Разумеется, моделировать на компьютере можно далеко не все явления, связанные с русским ударением. И все же есть немало случаев, когда для отдельных групп слов правила просты, а число исключений невелико. Вот тут-то компьютерное моделирование и уместно и полезно.

Во время нашей беседы мы несколько раз столкнулись с существительными, оканчивающимися сочетанием *-ент: цеме́нт, докуме́нт, доце́нт, парла́мент, регла́мент*. Я как раз недавно вывел правило о месте ударения в таких словах. Вот оно: «Существительные на *-амент* имеют в начальной форме ударение на втором слоге от конца (*орна́мент, перга́мент* и т. п.); исключения: *медикаме́нт, постаме́нт;* прочие неодносложные существительные на *-ент* имеют ударение на последнем слоге (*цеме́нт, инструме́нт, проце́нт* и т. п.); исключение: *ре́гент*».

Прошу заметить, что на самом деле число исключений несколько больше, чем в моей формулировке, но я сознательно опустил редкие, специальные слова. Попробуем построить алгоритм этого правила.

А л е ш а. Готово! Вот алгоритм:



Л е к т о р. Превосходно! Где же ты так хорошо научился строить алгоритмы?

А л е ш а. Читаю «ИНФО».

Л е к т о р. А теперь построим компьютерную модель нашего правила. Лингвистическое моделирование удобно проводить с помощью специальной моделирующей системы, которая называется...

А л е ш а. СИМВОЛ!

Л е к т о р. Да, но откуда ты знаешь о ней?

А л е ш а. Читаю «ИНФО»[5].

Л е к т о р. В таком случае ждем от тебя модели, понятной для СИМВОЛа.

А л е ш а. Я составил модель по алгоритму:

ПРАВИЛА	МОДЕЛЬ
1. МЕДИКАМЕНТ →	МЕДИКАМ′ЕНТ
2. ПОСТАМЕНТ →	ПОСТАМ′ЕНТ
3. АМЕНТ →	′АМЕНТ
4. РЕГЕНТ →	Р′ЕГЕНТ
5. ЕНТ →	′ЕНТ

Л е к т о р. Включаю компьютер, вызываю СИМВОЛ, ввожу твою модель. Вам, друзья мои, предоставляю набрать интересующие вас слова на *-ент*. Готово? Ну а теперь посмотрим, как компьютер поставит в них ударение.

М. И. Что я вижу? Ужасно! Это издевательство над русским языком — языком Пушкина и Толстого! Что означают эти два значка ударения в словах МЕДИКАМ″ЕНТ и ПОСТАМ″ЕНТ? Почему в словах ОРН′АМ′ЕНТ и Р′ЕГ′ЕНТ два ударения? Нет, я недаром всегда с подозрением относилась к компьютеризации, особенно в нашем предмете. Словеснику компьютер не нужен!

Лектор. Успокойтесь, любезнейшая Мария Ивановна. Алеша допустил ошибку в составлении модели; сейчас он ее найдет и исправит.

Алеша. Я уже понял, в чем дело. Слово МЕДИКАМЕНТ подпало сначала под правило номер один, а затем еще и под правило номер пять. То же самое случилось и с другими словами, которые взволновали Марию Ивановну: ПОСТАМЕНТ (правила 2, 5); ОРНАМЕНТ (правила 3, 5); РЕГЕНТ (правила 4, 5). Но я не знаю, как исправить ошибку: ведь если я переставлю куда-нибудь пятое правило, то компьютер неправильно поставит ударение во многих других словах. Какой-то замкнутый круг...

Лектор. Ну, во-первых, я бы посоветовал тебе ставить знак ударения не перед, а после буквы, обозначающей ударный гласный, как это принято в некоторых словарях. Тогда слова МЕДИКАМЕНТ и ПОСТАМЕНТ после их обработки компьютером не будут казаться странными, ведь пятое правило к ним не применится. Ну а дальше...

Сидоров. Мне кажется, я придумал, что надо сделать. Давайте как-нибудь пометим слова, которые, пройдя обработку по одному из правил, уже не должны подпадать под другое. А в конце мы с них снимем помету. Я предлагаю такой вариант модели:

1. МЕДИКАМЕНТ → МЕДИКАМЕ′НТ
2. ПОСТАМЕНТ → ПОСТАМЕ′НТ
3. АМЕНТ # → А′МЕНТ * #
4. РЕГЕНТ # → РЕ′ГЕНТ * #
5. ЕНТ # → Е′НТ #
6. * → ␣

Лектор. Пока мы спорили, вы, оказывается, успели настолько понять СИМВОЛ, что даже освоили знак конца слова (#). Признаться, даже для меня ваше решение неожиданно. Значит, слово, не нуждающееся в применении пятого правила, получает звездочку на конце. Так... а пятое правило применяется только к словам, где ЕНТ — чисто конечное сочетание. Любопытно проверить вашу модель.

Алеша. Я уже проверил. Все правильно.

М. И. Нет, не все. Я ввела в компьютер слово АПАРТАМЕНТ и получила, как вы говорите, «на выходе» АПАРТА′МЕНТ. Но ведь я знаю, что надо говорить АПАРТАМЕ′НТ.

Лектор. Увы, это как раз тот случай вариантности, с которым компьютер справиться не может. Словари здесь дают противоречивые сведения, однако все больше распространяется и признается именно то ударение, которое поставил компьютер. Это и понятно: слово перестает быть исключением и подчиняется правилу.

Сидоров. А у меня возникло сомнение по поводу слова МЕНЕДЖМЕНТ. Компьютер поставил ударение на последнем слоге, но я никогда не слышал, чтобы это слово так произносили.

Лектор. Вы правы. Это существительное произносится с ударением на первом слоге: МЕ′НЕДЖМЕНТ. Это, конечно же, исключение. Я не предусмотрел его в своей формулировке правила, потому что его нет еще в словарях.

М. И. Как я понимаю, если учесть в модели все исключения, то у нас получится своеобразный компьютерный справочник ударения существительных на *-ент*. Это, по-моему, очень удобно. Нет, все-таки в компьютерном моделировании что-то есть...

Сидоров. А я хотел бы знать, для каких еще групп слов можно создать подобные справочники.

Лектор. Поскольку наше машинное время ограничено, назову только некоторые группы имен существительных, которые часто требуют справок об ударении. Вот, например, слова на *-лог*. Я покажу вам алгоритм правила, по которому вы сможете и «восстановить» формулировку, и построить компьютерную модель.

М. И. А ведь это прекрасное задание! Я непременно предложу его своим ученикам.

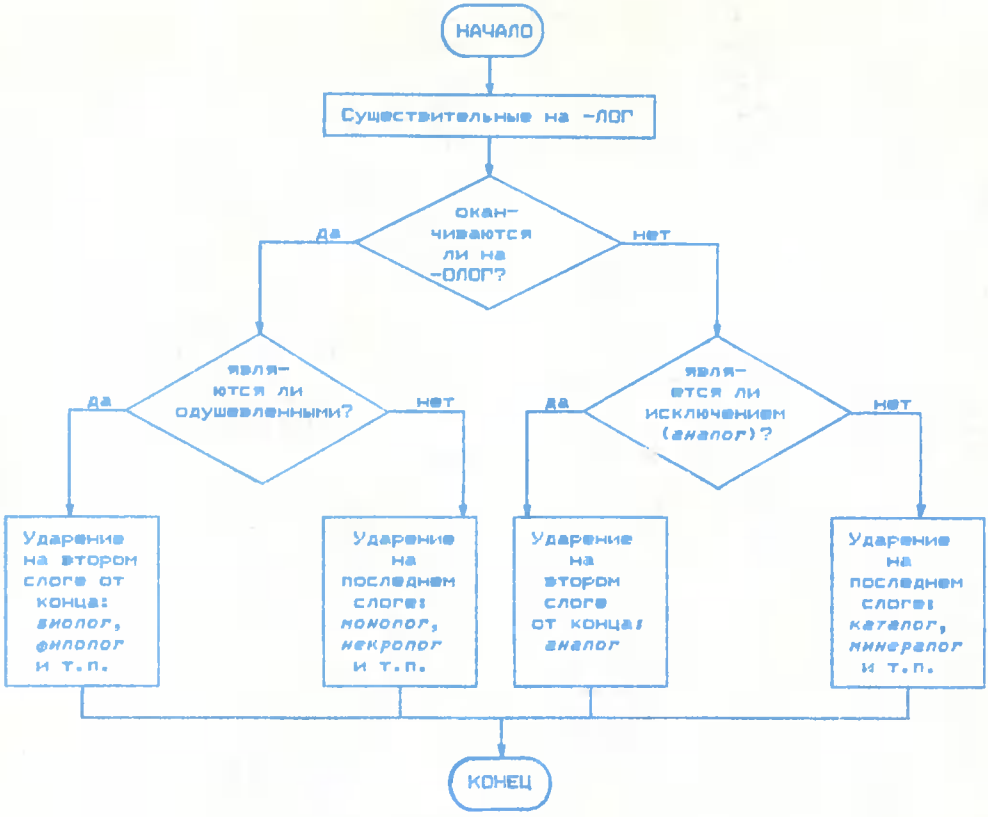
Лектор. Итак, перед вами алгоритм, позволяющий определить место ударения существительных на *-лог* (см. след. стр.).

Алеша. Не сможет компьютер работать по этому алгоритму: как он отличит одушевленное существительное от неодушевленного?

Лектор. Конечно, для компьютера нужно задать списком одну из этих групп, лучше — неодушевленные существительные, потому что их немного: *аполог*, *мартиролог*, *монолог*, *некролог*, *пролог* (вступление; не путать со словом *пролог* — церковная книга).

М. И. Интересно, что все слова из этого списка означают тексты или часть текста.

Лектор. А сейчас я предложу вам компьютерную модель, позволяющую обозначать ударение в словах на -МЕТР. Вы же, надеюсь, сможете самостоятельно сформулировать правило и построить алгоритм. Думаю, что Мария Ивановна и на этот раз объединит



«списочные» слова по значению. Итак, представляю:

МОДЕЛЬ

ОБОЗНАЧЕНИЕ

S:М Р С Т (в данном случае все согласные, возможные перед -МЕТР).

ПРАВИЛА

1. (S) МЕТР # → МЕ′ТР # (вольтме́тр)
2. МИЛЛИМЕТР # → МИЛЛИМЕ′ТР #
3. САНТИМЕТР # → САНТИМЕ′ТР #
4. ДЕЦИМЕТР # → ДЕЦИМЕ′ТР #
5. КИЛОМЕТР # → КИЛОМЕ′ТР #
6. ГРАММОМЕТР # → ГРА′ММОМЕ′ТР # (килогра́ммоме́тр)
7. КУБОМЕТР # → КУБОМЕ′ТР #
8. АМЕТР # → А′МЕТР # (вольта́метр)
9. ИМЕТР # → И′МЕТР # (пери́метр)
10. ОМЕТР # → О′МЕТР # (динамо́метр)

Л е к т о р. Теперь обратите внимание вот на эту распечатку:

БАГАЖ → БАГАЖА′
ПЕЙЗАЖ → ПЕЙЗА′ЖА
ЛИСТАЖ → ЛИСТАЖА′
ЭТАЖ → ЭТАЖА′
ТРИКОТАЖ → ТРИКОТА′ЖА

С и д о р о в. Ну уж это слишком. Не подделка ли? Компьютер образует форму родительного падежа и именно в ней обозначает ударение. Помню, в одном из выступлений я сказал, что необходимо увеличить производство трикотажа́, а в ответ раздался смех. Неужели здесь правила столь просты, что их мог бы подсказать мне компьютер?

А л е ш а. Вы предлагаете нам добиться результата, который показали. Но ведь для этого сначала нужно вывести правила из материала, а где его взять?

М. И. Для того чтобы найти в словарях все существительные на *-аж*, нужно очень много времени. А у меня еще факультатив, а тетрадей, тетрадей!

Л е к т о р. Но если вы возьмете «Грамматический словарь русского языка», составленный А. А. Зализняком, то долго искать существительные на *-аж* вам не придется: в этом словаре слова расположены по алфавиту конечных буквосочетаний. Рядом с каждым условно обозначена схема ударения (подвижное или неподвижное). Правила ударения слов на *-аж* осложняются исключениями и вариантами. Выбрать для модели предпочтительный вариант вам помогут орфоэпические словари.

С и д о р о в. Трудное задание!

Лектор. Начните с более простой модели — ударения существительных на *-граф* (типа *гео́граф, каллигра́ф*).

М. И. Это очень интересно! Ведь еще в прошлом веке ударение в этих словах отличалось от современного. Помните, у Пушкина в «Евгении Онегине»:

Он знал довольно по-латыни,
Чтоб эпигра́фы разбирать...

Лектор. Наша беседа окончена. Желаю вам успехов.

Эпилог

Прошло две недели. Перед началом следующей встречи с Лектором собрались уже знакомые нам слушатели.

М. И. Я просмотрела слова на *-граф* и обнаружила, что ударение у них на втором слоге от конца: *био́граф, пара́граф, эпи́граф.* Исключений немного: *каллигра́ф, полигра́ф, телегра́ф* да еще два-три специальных.

Сидоров. Но, кроме этого, ударение на последнем слоге имеют слова, означающие графские титулы: *бургграф, ландгра́ф, маркгра́ф.* Я их тоже учел в своей модели.

Алеша. А я сформулировал правило для слов на *-метр:* в названиях единиц измерения ударение на последнем слоге *(милли́метр, кубоме́тр)*; в прочих случаях, если перед *-метр* находится согласная, ударение также на последнем слоге (*вольтме́тр, оммéтр*), а если гласная — на предпоследнем (*вольта́метр, динамо́метр*). Слова *гео́ме́тр* и *дози́ме́тр* имеют варианты ударения. Я выбрал те из них, которые соответствуют общему правилу: геóметр, дозúметр.

М. И. Признаться, я очень увлеклась работой со словарями и составлением алгоритмов, и, кажется, эта моя увлеченность передалась и ученикам. Ведь это такой новый и необычный для них взгляд на язык! Только со словами на *-аж* у меня не очень получается.

Сидоров. Я понял только, что если в таких словах более двух слогов и нет второстепенного ударения, то они имеют неподвижное ударение. Теперь в своих выступлениях я говорю: *производство трикота́жа, учет киломе́тра́жа, осуществление демонта́жа.*

Алеша. Да, правила ударения в русском языке зависят от многих причин. Но главное — ударение все-таки подчиняется правилам и его может обозначать в словах компьютер!

1. Словарь ударений для работников радио и телевидения. 5-е изд. М., 1984.

2. Орфоэпический словарь русского языка. М., 1983.

3. *Зализняк А. А.* Грамматический словарь русского языка. М., 1977.

4. *Зализняк А. А.* От праславянской акцентуации к русской. М., 1985.

5. *Манзюра В., Шмелев Д.* Система для учебного моделирования языковых явлений // Информатика и образование. 1989. № 1.

Н. ШЕРПАЕВ
г. Брянск

Взгляд на олимпиаду

Олимпиады по информатике стали традицией. Результаты столичных, всесоюзных и международных олимпиад существенным образом отражают состояние развития школьной информатики в передовых школах нашей страны. А каково же положение дел в основной массе школ? Типичным примером, отражающим этот уровень, может служить областная олимпиада в Брянске.

В Брянской области насчитывается около 450 средних общеобразовательных школ и профессионально-технических училищ, около 90 кабинетов вычислительной техники (от 5 до 14 ПЭВМ в каждом). Любопытно, что кабинеты области снабжены комплектами вычислительной техники не менее десяти наименований.

Олимпиада, прошедшая с 30 января по 1 февраля 1990 г., была второй олимпиадой по информатике, проведенной Брянским ИУУ. В первой (1989) участвовали преимущественно учащиеся, направленные школами Брянска. В нынешнем году предварительно были проведены школьные и районные олимпиады по всей области. Для проведения областной олимпиады была создана методическая комиссия, в состав которой входили преподаватели пединститута, лучшие учителя школ.

Важным этапом подготовки олимпиады была выработка концептуального подхода для разработки и оценки заданий. Была сочтена целесообразной разработка заданий, отражающих учебный материал курса информатики и направленных на выяснение уровня знаний, умений и способностей учащихся.

При обсуждении содержательной стороны заданий было решено подготовить задания для разработки разнообразных по характеру алгоритмов, считать вполне доступным включение задач по программированию и нестандартных задач по математике, задач нематематического характера, на использование педагогических программных средств, на кодирование и декодирование.

Следуя установившейся в стране традиции, олимпиада проходила в два тура: теоретический и практический.

При проведении теоретического тура выяснялись способности учащихся ориентироваться в предложенной ситуации, формализовать поставленные задачи, создавать, корректировать и оценивать алгоритмы решения задач.

Для проведения практического тура была осуществлена дифференциация учащихся. Учащиеся были разделены на группы в соответствии с классом обучения, уровнем знакомства с компьютером, адресом проживания. С учащимися групп были проведены соответствующие консультации по использованию ВТ. Дополнительные консультации давались учащимся в ходе решения ими задач. На практическом туре от учащихся требовалось осуществление всех этапов решения задач, начиная от постановки, включая выбор языка программирования, отладку и выход на ЭВМ.

Теоретический тур

1. Составить на алгоритмическом языке алгоритм, определяющий, сколько точек с целочисленными координатами лежит внутри окружности с центром в точке с координатами (X, Y) и радиусом R.

2. Дана таблица B, состоящая из N столбцов и M строк и содержащая различные вещественные числа. Составить на алгоритмическом языке алгоритм, меняющий местами наибольший и наименьший элементы таблицы.

3. Даны две линейные целочисленные таблицы X и Y длиной N и M соответственно, причем элементы таблицы X расположены в порядке убывания, а Y — в порядке возрастания. Составить алгоритм, формирующий из X и Y линейную, упорядоченную в порядке возрастания таблицу Z.

4. Даны четырехугольник ABCD и точка M(X, Y). Составить алгоритм, определяющий, находится ли заданная точка в четырехугольнике; его вершины заданы координатами: A(Xa, Ya), B(Xb,Yb), C(Xc, Yc), D(Xd, Yd).

5. Дана литерная величина A, содержащая несколько пар открывающих и закрывающих круглых скобок. Составить алгоритм, удаляющий части текста, заключенные в скобки.

Практический тур

1. Составить программу для решения уравнения

$$160*X\wedge 3-372*X\wedge 2-76*X+15=0.$$

2. Составить программу, моделирующую движение шарика в прямоугольной коробке. Начальные данные: координаты шарика (Xo, Yo) и d — угол наклона вектора скорости к горизонтали (в градусах). Соударения со стенками абсолютно упругие. Угол падения равен углу отражения.

3. «Считалка». Даны натуральные числа N и M. Предполагается, что N человек встают в круг и получают номера против часовой стрелки: 1, 2, 3,..., N. Затем, начиная с первого, также против часовой стрелки отсчитывается M-й человек (поскольку люди стоят по кругу, то за N-м человеком стоит первый). Этот человек выходит из круга, после чего начиная со следующего снова отсчитывается M-й человек, и так до тех пор, пока из всего круга не останется один человек. Составить программу для определения его номера.

Для оценки решений приведенных заданий использовались следующие критерии: правильность решения задачи, уровень разработанности алгоритма, эффективность решения, учет граничных и особых решений, ясность описания алгоритма. Учитывалась трудность задачи.

Анализ работ позволил выявить особенности и недостатки в подготовке учащихся и слабые стороны в организации олимпиады. Особые затруднения учащиеся испытывали в решении нестандартных задач, в использовании языков программирования. К особенностям можно отнести нежелание учащихся использовать алгоритмический язык действующего учебника информатики. Следует отметить недостаточную активность в подготовке учащихся к олимпиаде со стороны школ, имеющих классы с углубленной математической подготовкой, Брянского филиала ВЗМШ, школьного клуба «Компьютер», факультета дополнительных педагогических профессий пединститута.

Вниманию руководителей предприятий, совместных предприятий, сервисных организаций, кооперативов!

По «Орловский завод УВМ им. К. Н. Руднева» освоил производство тестера цифровых и аналоговых блоков КД-1, позволяющего осуществлять:

▲ поиск неисправностей в блоках с комбинаторной логикой, в блоках ОЗУ, ПЗУ и аналого-цифровых блоках;

▲ проверку аналого-цифровых блоков.

Тестер КД-1 является средством измерения и включен в Госреестр СССР.

Тестер КД-1 имеет 288 логически программируемых дискретных входов-выходов или 84 аналоговых и 64 дискретных входа-выхода.

Частота задания воздействий — от 10 до 10 000 кГц.

Вид контроля — функциональный и параметрический.

Среднее время проверки — 2 мин.

Имеется язык высокого уровня для написания тестов проверки блоков.

Тестер комплектуется стандартным стыковочным узлом для подключения плат типа Е2.

Тестер позволяет контролировать электронные блоки от любых ЭВМ, персональных ЭВМ отечественного и зарубежного производства, блоки от станков с ЧПУ.

Объединение принимает заказы на разработку по отдельному договору стыковочных узлов, программ проверки блоков, программ ремонта блоков.

Наш адрес:
302025 Орел, т. а. Нейтрон 148216 и 148234; тел. 3-15-31.

К сведению проектировщиков и пользователей

Впервые в СССР ПО «Орловский завод управляющих вычислительных машин им. К. Н. Руднева» приступает к освоению интегрированной комплексной системы автоматизации рабочих мест — системы **«ИКАР».**

Идеология системы **«ИКАР»** исходит из необходимости обеспечения возможности разработки систем автоматизации различных направлений — от учебных классов до систем массового обслуживания, от простейших АСУ технологическими процессами в сельском хозяйстве до АСУ ТП сложных производств типа химических, от конторских АРМ до САПР проектирования БИС — на основе использования:

▲ единой операционной среды МДСС, совместимой с MS DOS 3.3;

▲ унифицированных аппаратных средств типа IBM PC;

▲ полной переносимости программного обеспечения, наработанного ранее для IBM PC;

▲ выносных устройств сопряжения с объектом (УСО) трех типов, позволяющих автоматизировать практически любые производства;

▲ аппаратных и программных сетевых средств;

▲ программных продуктов, поставляемых по заказу пользователя;

▲ сервисного оборудования.

В состав системы «ИКАР» входят:

▲ 8-разрядная универсальная ЭВМ;

▲ две 16-разрядных ЭВМ, предназначенные для управления и построения АРМ;

▲ три типа УСО — МСКУ, МКСО, С601М;

▲ сервисное оборудование для ремонта и метрологической проверки электронных блоков.

Наш адрес:
302025 Орел, т. а. Нейтрон 148216 и 148234; тел. 3-15-31.

ЧТО МОЖЕТ ЭВМ?

Прометей XX века

Возможно, вы уже догадались, что речь пойдет об «электронном» «Прометее». Так называется западноевропейская исследовательская программа, цель которой — повысить с помощью электроники безопасность езды на автомобиле. В ее рамках создаются компьютерные системы управления движением, бортовые компьютеры, сенсорные устройства для различных компонентов автомобиля, уличных светофоров, дорожных указателей и т. д.

Компьютеры могут управлять жесткостью подвески, положением колес при повороте, тормозами — все это уже используется на серийных машинах и позволяет, например, не опасаться «юза» даже на большой скорости или плохой дороге. Экзотические пока системы управления голосом также увеличивают безопасность, позволяя меньше отвлекаться, например, на включение стеклоочистителей. Впрочем, последние могут включаться автоматически — компьютер способен определить, когда начинается дождь.

Другая область использования ЭВМ для повышения безопасности — моделирование. Уже сократилось до двух часов время, необходимое для имитации разрушений, происходящих при аварии автомобиля. Стоимость таких расчетов велика, но и натуральные испытания недешевы, а главное, компьютер позволяет «испытывать» автомобиль на начальной стадии проектирования, задолго до изготовления первого экземпляра. Так удается избежать трудноустранимых впоследствии ошибок конструкторов.

Новые возможности дает ЭВМ для организации «интерфейса» автомобиля и водителя. С ее помощью можно улучшить приспособленность как автомобиля к человеку (путем автоматической индивидуальной подстройки всех параметров — от положения водительского кресла до характеристик тормозной системы и коробки передач), так и человека к автомобилю (используя тренажеры для отработки водительских навыков).

Сокращение времени реакции водителя на опасность хотя бы на полсекунды может предотвратить половину автокатастроф, считают специалисты. По всей видимости, компьютеризация поможет «сэкономить» эти мгновения. Но не случится ли так, что, надеясь на ЭВМ, люди будут менее осторожны за рулем и число катастроф останется на прежнем уровне?

Все же главное — человек.

Плейер с памятью или компьютер с музыкой?

Японская фирма Korg начала продавать компактный плеер, работающий под управлением программ, записанных на стандартных картах памяти. Размером всего 161×100×33 мм, он снабжен жидкокристаллическим дисплеем, одноклавишным переключателем программ, может цифровым способом создать эхо-эффект и эффект присутствия и, разумеется, имеет автостоп. Плеер, позволяющий запомнить на карте памяти до 40 музыкальных программ, стоит 56 долл., а 16-программная версия — всего 34 долл.

Контакт обеспечит резина

Резина всегда считалась хорошим изолятором. Недаром электрики работают с высоким напряжением, стоя на резиновом коврике и в резиновых перчатках. Но, как это ни парадоксально, на основе именно этого эластичного материала японские конструкторы создали целый класс различных упругих контактов и гибких мембранных клавиатур.

Основным достоинством резиновых электрических кнопок являются простота и технологичность изготовления, а также полная герметичность клавиатур, изготавливаемых на их основе. Более того, резиновые кнопки не боятся работать в агрессивных средах и весьма вибростойки.

Кроме того, на основе резиновых контактов клавиатура значительно надежнее привычных «жестких» образцов, поскольку в ней практически полностью отсутствуют механические детали. А применение разноцветных материалов и точного литья предоставляет широкий простор дизайнерам.

Словосочетание «резиновый контакт» для нас пока еще непривычно, хотя этот материал уже посещает наш быт. Например, тонкая пластинка жидкокристаллического индикатора в наручных электронных часах подключается к контактам кристалла с помощью полоски именно такого экзотического материала. Так что теперь, прежде чем доверить свою жизнь резиновому изделию, проверьте его на электропроводность.

Погода с доставкой на дом

Имея модем и ПЭВМ «Макинтош», вы можете подключиться к базе данных Accu-Data Национальной метеорологической службы США. В нее стекаются данные, ежечасно фиксируемые метеостанциями на всей территории страны. Разобраться в них поможет программа Accu-Weather Forecaster.

Задав район на карте США, вы получите полную информацию о его погоде сейчас или в прошлом: температуру, давление, влажность, направление и скорость ветра, высоту и плотность облачности... Все это отображается в виде текста или метеокарты; можно заказать построение вертикального разреза атмосферы до высоты 5000 м. При некотором умении (предыдущие операции доступны даже дилетанту) можно получить также информацию о погоде вне территории США.

Стоит все это недорого: кроме цены программы (89 долл.) нужно оплатить подключение к БД Accu-Data (40 долл. плюс 10 долл. ежемесячно) и работу с ней (18 центов в минуту ночью, 38 — днем; загрузка данных занимает обычно 3—4 мин), а также аренду канала связи (18 центов в минуту для американских абонентов).

А. БОРК
Центр педагогических технологий,
Калифорнийский университет

Компьютеры в обучении: чему учит история

История компьютеризации обучения во многих отношениях вселяет чувства, далекие от оптимизма. Похоже, на чужих ошибках учатся мало. Общая тенденция состоит в том, что новая страна или группа (в пределах одной страны) начинают с таких форм, которые уже были кем-то опробованы и отвергнуты. Вероятно, ошибки ранних этапов необходимы для дальнейшего развития. И все же можно надеяться, что знакомство с существующим опытом позволит избежать их постоянного повторения.

Хотелось бы предупредить читателя, что последовательность этапов компьютеризации, которой автор придерживается при изложении материала, не следует воспринимать как жесткую схему. В каждой конкретной ситуации имеются свои особенности.

Начало

В этом разделе обсуждаются начальные этапы внедрения компьютеров (и связанной с ними технологии) в обучение. Описываемую схему можно обнаружить и на уровне отдельной школы, и на уровне страны в целом.

«Достанем побольше техники»

Стремление к приобретению и установке в школах и университетах возможно большего числа компьютеров — наиболее ярко выраженная тенденция, которую мы обнаруживаем на начальном этапе. Внимание главным образом сосредоточено на приобретении техники, хотя иногда отмечается и стремление дать учителям соответствующую подготовку.

На этом этапе учителя и администрация школ руководствуются желанием расширить кругозор, но главными движущими силами, по-видимому, являются родители и деловой мир. Интересы делового мира достаточно очевидны — стремление обеспечить рынок сбыта. Родители же руководствуются смутным представлением о том, что если школа или университет не имеют компьютеров, то здесь что-то не так. Обычно никто из перечисленных действующих лиц не имеют ясного представления о том, для чего нужны компьютеры.

Иногда, правда, имеется в виду некоторый — единственный — способ применения компьютеров. Это более продвинутая, но все же ориентированная на технику стадия. Ее можно охарактеризовать так: «Закупим много компьютеров, работающих с программой X, подготовим учителей, и наша школа преобразится». Однако надежды, связанные с подобным программным обеспечением, не оправдываются, а сами компьютеры фактически не воздействуют на жизнь учащихся.

Подобно многим другим, эта стадия не уходит бесследно, а повторяется снова и снова. Даже сегодня можно наблюдать примеры приобретения компьютеров без программного обеспечения или с одним-единственным его видом. На этой стадии никак не учитываются содержание и закономерности обучения, эти важные факторы, к которым мы еще будем обращаться.

Каковы здесь исторические аналогии? Представим, что на заре книгопечатания школы приобретали бы печатные станки, надеясь, что теперь-то все пойдет по-другому. Можно также вообразить школу, в классах которой установили видеомагнитофоны, но не позаботились о видеозаписях: не выяснили, какие записи имеются, как их получить, какую функцию они должны выполнять в учеб-

ном процессе. В этих примерах технические аспекты вытесняют педагогические на второй план. То же происходит, если акцент делать только на приобретении компьютеров.

В начальных и средних классах государственных школ США установлено около двух миллионов компьютеров. Возникает вопрос: разумно ли было покупать столько компьютеров, учитывая ограниченную эффективность их применения в учебном процессе? Нет никакой необходимости, чтобы все шли по этому пути. В Японии, стране с развитой технологией, прилагаются сознательные усилия не допустить проникновения компьютеров в школьные классы, по крайней мере на ранних этапах. В некоторых других странах также пытаются избежать этого, но в основном потому, что компьютеры слишком дороги или имеются сложности с их производством. Может быть, странам этой последней группы повезло!

«Будем обучать языкам»

После того как техника приобретена, во многих школах наблюдается замешательство. Как ее использовать? Что делать учащимся? Положение отчаянное.

Один тип программного обеспечения доступен всегда — это простые языки программирования. Они обычно поступают вместе с компьютером, и типичное применение полученных машин — обучение программированию. Изучение языка программирования — это, как правило, не то, что имели в виду родители, учителя и администрация школы. Но такая возможность существует, и как часто бывает при использовании технологии, то, что возможно, происходит, хотят этого или нет. Таким образом, учащиеся усваивают языки программирования, а не более важный учебный материал.

Исходя из посылки, что учить программированию нужно, подыскиваются соответствующие аргументы: «в будущем программировать придется практически всем», «разобраться в компьютерах можно, лишь создавая программы», «программирование развивает способность к решению задач». Эти предположения сомнительны, неопределенны и не имеют достаточного экспериментального подтверждения (хотя третье из них при соответствующем содержании обучения вполне может оказаться справедливым).

Разумеется, программирование полезно для некоторых людей, а может быть, и для всех, однако это еще нужно показать. Многие сторонники обучения программированию связывают его с освоением общих стратегий решения задач. Однако исследования в этой области не обнаружили ожидаемого, решение задач в одной области часто не связано с решением задач в другой области. Тем не менее здесь имеются интересные возможности, которые могут быть реализованы в будущем.

По моему мнению, для обучения программированию должны быть созданы гораздо более интересные учебные среды. Примеры некоторого продвижения в этом направлении — разработка «Робота Мартино» в Италии и материалы по курсу информатики, подготовленные Академией наук СССР и Московским государственным университетом. Каждый из этих проектов ориентирован на целостный учебный курс, а не на чистое обучение языкам.

Вопрос о том, следует ли обучать программированию (который распадается на подвопросы: как обучать, кого и когда), остается по меньшей мере открытым. Когда программированию учат в школе, обычно сталкиваются с тремя серьезными проблемами. Во-первых, отсутствуют соответствующие учебные материалы. Во-вторых, школьные учителя мало знакомы с современным программированием. В-третьих, при обучении школьников применяют далеко не лучшие языки.

Не останавливаясь на нехватке учебных материалов по программированию, отмечу, что большинство учебников написаны людьми, не имеющими достаточного опыта в этой области. Еще больше это касается учителей. Современное программирование — непростая деятельность. Принципы разработки программного обеспечения не могут быть освоены новичками во время летних курсов. Никому и в голову не приходит, что за несколько недель можно сделать из новичка учителя математики. В отношении программирования это считается допустимым, хотя хорошего учителя программирования подготовить не легче, чем хорошего учителя математики.

Вопрос о языках следует рассмотреть подробнее. Здесь также имеет место история, повторяющаяся в различных вариантах. Почти неизбежно первым языком является Бейсик, хотя ситуация понемногу меняется. Мало кто из специалистов верит, что обучение Бейсику имеет какой-то смысл. Бейсик создан давно и не удовлетворяет многим появившимся позднее принципам разработки программного обеспечения. Он способствует формированию вредных навыков, которые в дальнейшем крайне трудно преодолеть. Все это хорошо известно, однако Бейсик по-прежнему остается широко распространенным учебным языком.

Не думаю, что ситуация значительно улучшилась с появлением в школах Лого. Он

позволяет структурировать программы (за счет использования процедур) и в этом превосходит многие варианты Бейсика, однако по существу это старый язык со странным по сравнению с более поздними языками синтаксисом. В 99 % случаев учащиеся имеют дело не с фундаментальными идеями и средствами Лого, а с графическими возможностями, которые дополнили этот язык уже после его разработки. Черепашью геометрию можно добавить практически к любому языку. Большинство учителей не понимают Лого и соответственно учат школьников. Поэтому и результаты, как правило, невысоки.

Экзамен «Advanced Placement»* по информатике способствовал выбору более адекватного языка программирования, но не затронул описанных выше фундаментальных проблем. Этот экзамен требует знания структурного языка Паскаль и может быть зачтен в колледже. Выбор Паскаля не случаен — более 95 % вводных университетских курсов по информатике основаны на этом языке.

Использование компьютеров для обучения языкам следует рассматривать и в более широком контексте. Школьники изучают алгебру, физику, английский язык, историю и другие предметы. Если компьютер не будет применяться при их изучении, маловероятно, чтобы он существенно повлиял на систему образования. Следовательно, обучение языкам, даже очень успешное, отражает лишь ограниченное использование компьютера в школе. Похоже, среди специалистов существует согласие по этому поводу, однако языкам продолжают обучать, и зачастую плохо. И все же я считаю вопрос о целесообразности обучения языкам открытым.

«Будем обучать компьютерной грамоте»

Этот этап чаще всего следует за двумя только что описанными. Нередко сначала это всего лишь другое название (для многих более приемлемое) занятий по программированию. Аргументы здесь подобны тем, что выдвигаются в пользу обучения языкам: компьютеры будут широко использоваться в различных сферах деятельности; для полноценной жизни в обществе будущего каждый должен быть компьютерно грамотным. Мнения родителей, учителей и администрации при этом чаще всего совпадают.

Однако согласия нет, когда речь заходит о содержании курса компьютерной грамотности, которая хоть и приветствуется всеми, но не имеет четкого и ясного определения. Люди соглашаются, что она нужна лишь потому, что не достаточно задумываются о смысле, который вкладывают в этот термин.

Компьютерная грамотность — это своего рода связующее звено между ранними и последующими этапами компьютеризации обучения. На одном из этапов компьютерная грамотность практически совпадает с обучением элементарным основам программирования. И действительно, некоторые учебники компьютерной грамотности являются, в сущности, учебниками программирования, и здесь справедливы замечания к предыдущему разделу.

Компьютерную грамотность определяют и по-другому. Например, на самом примитивном уровне это может означать: «я могу включить компьютер, вставить дискету и запустить программу». Может подразумеваться также использование таких средств, как электронные таблицы или системы подготовки текстов.

Важный аспект компьютерной грамотности, который учащимся бывает трудно разъяснить,— понимание этических и моральных проблем, связанных с использованием компьютеров. Эти машины оказывают на общество огромное влияние. Если не проявлять достаточной осмотрительности, можно столкнуться с серьезными социальными, экономическими и личностными трудностями. Однако в большинстве курсов компьютерной грамотности этот аспект если и затрагивается, то поверхностно.

Мои наблюдения в школах, где, как утверждается, введен курс компьютерной грамотности, показали, что часто он сводится к компьютерным играм. Это объясняют по-разному, и в первую очередь тем, что игры помогают учащимся преодолеть психологический барьер к компьютеру. Откровенно говоря, я не замечал такого барьера, особенно у детей; хорошие дружественные компьютерные программы не нуждаются в попытках убедить учащихся, что компьютер — это хорошо. Утверждается также, что компьютерные игры полезны, так как они увлекают учащихся. Однако равным образом можно доказывать, что в классах полезны машины для игры в пинбол*, бесплатные сладости или эротика.

К концу описываемого этапа термин «компьютерная грамотность» становится непопулярным. Иногда вместо него используют другие. В Европе получил распространение термин «информатика», хотя он часто обозначает курс более высокого уровня.

* Экзамен, определяющий исходный уровень подготовки студентов.

* Китайский бильярд.

Продолжение

С этого момента наша история становится более неопределенной: не всегда наблюдаются все перечисленные ниже этапы, может меняться их последовательность. Нередко после неудач, характеризующих начальные этапы, направления компьютеризации начинает определять национальная политика.

Существуют различные мнения о том, какие этапы наиболее предпочтительны и какой порядок их прохождения является правильным, поэтому предложенный порядок не следует воспринимать как единственно возможный.

«Будем использовать передовую технику»

Тенденция к приобретению компьютеров была характерна для самого начала компьютеризации. Она проявляется и теперь, но уже на новом уровне — в стремлении приобрести более мощное оборудование.

Разновидность этой тенденции наблюдается в престижных университетах: закупают не просто необходимое оборудование, а самое современное и дорогое, которое только могут себе позволить. На его приобретение и разработку системного программного обеспечения расходуются большие средства. Вопросы обучения отходят на второй план, в большинстве случаев новая техника никак не отражается на студентах, а занятия проходят так же, как и прежде.

Другой аспект этого увлечения состоит в использовании новой технологии, связанной с компьютерами, в частности видеодисков и различных компакт-дисков. Некоторые группы мечутся от одного вида оборудования к другому в погоне за призраком.

Отсюда не следует, что подвергается сомнению польза этих устройств для обучения. Я уверен, они могут быть полезны, однако если все время гнаться за новинками, компьютер никогда не окажет существенного влияния на систему образования! И мы никогда не начнем использовать то, что уже имеем. Здесь мы снова сталкиваемся с чрезмерным увлечением техникой и недостаточным вниманием к педагогической стороне дела. Компьютер, пусть и с современной периферией, должен по-прежнему быть в центре внимания, так как именно он приводит к интерактивности и индивидуализации обучения.

Вопрос о том, какую технологию использовать, нельзя решать, исходя лишь из доступности последних технических достижений. Главную роль должны играть педагогические соображения. Решения, принятые без учета всего контекста обучения, скорее всего, будут неадекватными. Взять, например, интерактивное видео — комбинацию компьютера и видеодиска. В одних случаях зрительная информация крайне важна как непосредственно для усвоения учебного материала, так и для повышения мотивации, в других — не имеет большого значения.

«Будем разрабатывать небольшие программы для обычных курсов»

Такова реакция на описанную выше ситуацию, когда компьютер практически не воздействует ни на учебный процесс, ни на учащихся. Большинство учебных курсов соответствуют различным предметным областям. И если компьютеру суждено существенно повлиять на образование, он должен прийти в предметные области. Постепенно школы начинают понимать это.

И вот мы наблюдаем, как начинают появляться компьютерные учебные материалы. Как правило, на первых порах это «домашнее производство»: учитель, немного знакомый с программированием, пишет программу для своих учеников. Так поступают многие. Подобные программы невелики, так как время и возможности учителей ограничены. Затем создаются организации, которые собирают программы учителей, модифицируют их, совершенствуют и делают широко доступными, иногда на коммерческой основе. Впоследствии эти организации могут приступить к разработке собственных программ. Многие компании, распространяющие компьютерные учебные материалы, появились подобным образом.

а) Характеристики небольших программ

Их отличительный признак — малый объем. Они разрозненны, не охватывают значительного учебного материала, и работа с ними не занимает у учащихся много времени (обычно не более часа).

Учитывая все это, неудивительно, что они не оказывают большого влияния не предмет, в рамках которого используются. В традиционных курсах, основанных главным образом на лекциях и учебниках, появляются лишь вкрапления компьютерного обучения. По сравнению со временем, уходящим на слушание лекций или чтение книг, время работы за компьютером, как правило, невелико.

Другая проблема, часто возникающая в связи с малыми программами,— это контроль качества. Чаще всего они разрабатываются индивидуально, без соответствующих ресур-

сов и нередко людьми с небольшим опытом создания учебных материалов, поэтому неудивительно, что их качество низкое. Эти программы все же используются по уже указанным причинам: компьютеры стоят в школах, и нужно продемонстрировать их полезность, чтобы как-то оправдать затраты. Существуют и хорошие образцы, но они немногочисленны и занимают слишком мало учебного времени, чтобы оказать ощутимое влияние на учебный процесс.

С малыми программами связан еще один интересный, но негативный феномен. Широкое распространение получают программы, имеющие отдаленное отношение к содержанию обучения, понимают ли его в традиционном смысле или с точки зрения перспектив образования. Например, увлечение созданием с помощью компьютера различных плакатов, надписей или поздравительных открыток слабо связано с учебным процессом и отражает лишь наличие соответствующего программного обеспечения. В некотором смысле положение очень напоминает ситуацию с языками программирования. У нас есть, вот мы и пользуемся! Программное обеспечение существует, поэтому его используют, иногда интенсивно. В результате — отрицательное воздействие на учебный процесс: учебное время расходуется в ущерб важным составляющим обучения.

б) Типы малых программ

Их разработано несколько, и вопрос о том, какой наиболее предпочтителен, является предметом оживленных и нередко ожесточенных дискуссий. Эти обсуждения редко опираются на педагогические соображения или эмпирические данные о потребностях учащихся при изучении того или иного предмета. В их основе — просто набор расхожих представлений. Чего не хватает каждой стороне, так это основанных на опыте данных, подтверждающих ее позицию.

Одним из распространенных типов являются обучающие программы (tutorials). На самом элементарном уровне — это тренировочные программы, обеспечивающие практическое освоение материала учащимися. Натаскивание часто подвергается критике, однако и сегодня при обучении многим предметам оно является едва ли не центральным звеном учебного процесса. Например, повсюду в мире обучение математическому анализу включает большое число тренировочных упражнений по дифференциальному и интегральному исчислению. Разумно это или нет — другой вопрос, но вряд ли стоит критиковать тренировку с помощью компьютера, если она интенсивно используется при бескомпьютерном обучении. Компьютер здесь дает такие явные преимущества, как возможность незамедлительной обратной связи и помощи учащемуся.

Обучающие программы более высокого уровня представляют собой попытку воспроизвести некоторые аспекты деятельности индивидуального преподавателя. Лучшие из них обеспечивают индивидуализацию и активный характер обучения, способствуют повышению интереса учащихся. Группы, составленные из сильных преподавателей, могут успешно создавать такого рода программы. Однако в настоящее время их мало и для них характерны уже знакомые нам недостатки, связанные с малым воздействием на учебный процесс. Таким образом, данный этап компьютеризации имеет бóльший потенциал, однако он все еще не реализован.

Следующим типом малых программ, имеющим своих ревностных сторонников, являются моделирующие программы (simulations). Их называют по-разному. У нас в Ирвине* — «управляемыми мирами». Если они реализованы с помощью Лого, их называют «микромирами». Иногда используют термин «свободные среды». Суть этих программ проста. Компьютер имитирует некоторую часть реального или воображаемого мира, обеспечивая возможность «играть» с ним, воздействовать на него и наблюдать, что при этом происходит.

Для использования подобных программ имеются глубокие педагогические основания, которые не все понимают. Имитационное моделирование особенно полезно при решении такой важной задачи, как развитие интуиции в некоторой области, проницательности. Накопление индивидуального опыта является важным компонентом обучения, которым традиционные методики часто пренебрегают. Имитационные программы создают для этого богатые возможности.

Однако типичные малые имитационные программы обладают принципиальным недостатком. Они не являются интерактивными, не реагируют на действия или бездействие учащихся, и поэтому нельзя определить, приобрел ли учащийся интуицию или другие формы знаний об изучаемом процессе. Возникающие у учащихся затруднения во многих программах игнорируются. Поэтому имитационные программы оказываются полезными лишь для некоторых. Многим учащимся работать с ними скучно, так как они не понимают, что происходит и зачем они в этот процесс включены. Разработчики имитационных программ нередко предполагают помощь со стороны учителя, однако,

* Место расположения Центра педагогических исследований Калифорнийского университета.

учитывая число учащихся в типичном классе, в большинстве случаев это невозможно. Серьезной проблемой является и учебная мотивация. Часто имитационные программы гораздо более интересны самим разработчикам, чем учащимся!

Я называю имитационные программы, которые полностью игнорируют то, что делает пользователь, «голой» имитацией, так как они не имеют «одежды», необходимой для того, чтобы стать полноценным педагогическим продуктом. Эффективная имитационная программа должна контролировать, что знает и чего не знает учащийся, предлагать ему помощь и поддержку. Этими свойствами обладают немногие программы.

Недавно программам на имитационное моделирование появилась альтернатива, которая также связана с формированием у учащихся интуиции через расширение опыта. Речь идет о программах для лаборатории на базе микрокомпьютера. Идея состоит в непосредственном подключении к компьютеру измерительных приборов, фиксирующих различные аспекты внешнего мира — температуру, расстояние до объекта и т. д. Обычно такие программы генерируют графики зависимости измеренных (или вычисленных на их основе) показателей от времени. Пока имеется мало эмпирических данных, позволяющих сравнить этот подход с чистым моделированием. Многие ученые с предпочтением относятся к реальному лабораторному оборудованию, даже если учащийся, как и в случае компьютерной лаборатории, непосредственно не управляет им.

Подобно имитационным программам, большинство программ для компьютерных лабораторий не учитывают действий учащегося. Здесь также предполагается, что дополнительной информацией учащихся обеспечит учитель, и не учитывается, что он не сможет помочь всем.

Некоторые обучающие и имитационные программы называются «интеллектуальными», однако в этих программах редко используются методы искусственного интеллекта. Нам известно лишь несколько подобных программ, хотя их потенциал, безусловно, велик.

в) Элитарные программы

Для многих обучающих программ серьезная проблема состоит в том, что с ними могу работать только сильные учащиеся, а обычные быстро начинают испытывать затруднения. Учебные материалы такого рода заслуживают названия «элитарное программное обеспечение». Если учащиеся отобраны специально или преподаватели очень компетентны, недостатки «голой» имитации или других программ могут быть скомпенсированы, по крайней мере для некоторых учащихся. Однако даже в этих условиях вероятность восполнить упущенное для всего класса мала. Программное обеспечение такого рода часто описывают с восхищением, однако его применение в обычных условиях кончается неудачей.

«Воспользуемся авторской системой»

Желание писать небольшие программы быстро проходит, так как включенные в этот процесс учителя мало или ничего не знают о программировании. В поисках выхода обращаются к авторским языкам и системам. Учителям говорят: «Есть простой способ создавать учебные материалы, используя магическую силу компьютеров, который не требует никакого программирования». Это может сделать каждый, утверждает реклама.

Но с помощью авторских систем разработано мало качественного продукта. Если системой легко пользоваться, как это рекламируется, она позволяет составлять лишь простейшие учебные материалы. С более сложными труднее работать, но зато они обеспечивают более широкие возможности. К сожалению, авторские системы редко проектируют компетентные специалисты в области информатики, знающие, какими должны быть язык программирования или среда для разработки программ, поэтому эти системы обычно неадекватны задаче. Кроме того, они потребляют большое количество ресурсов и времени и в результате оказывают негативное влияние на компьютеризацию обучения — расходуются деньги, которые было бы лучше вложить во что-нибудь другое. Для большинства авторских систем конечный результат — доходы тех, кто их распространяет. Но, к сожалению, почти каждый новичок в области компьютерной педагогики убежден, что первым делом он должен выбрать авторскую систему или язык! Спрос сохраняется, и появляются все новые системы.

«Составим каталог программ»

Малых программ невысокого качества разработано множество, возникает необходимость составить их перечень. Существуют организации, которые готовят и распространяют такие списки. Неудивительно, что качество материалов, перечисленных в них, сильно варьирует. Если программа перспективна, ее могут приобрести коммерческим путем, тогда она не попадет в список. Таким образом, подобные списки, даже имеющие рубрикацию, редко бывают полезны.

«Оценим малые программы»

Следующий после каталогизации шаг — попытка оценить существующие программы. Оценка, конечно, желательна, так как дает представление о том, какие программы и в каких условиях будут полезны.

Но под оценкой понимают разные вещи. В данном случае имеется в виду «оценка коллеги» — программу оценивает учитель, который ее не писал.

Многие программы, получившие высокую «оценку коллеги», могут представлять серьезные трудности или быть непригодны для учеников, это характерно не только для компьютеров. Таким образом, хотя «оценка коллеги» может быть в чем-то полезной, она не позволяет оценить главное — работу с этой программой типичных пользователей. А комплексное оценивание, если его проводить качественно, в большинстве случаев требует много времени и средств.

Общий результат оценивания малых программ, даже если оно проводится по несовершенной методике, подтверждает вывод о том, что среди них очень мало учебных средств, эффективно воздействующих на процесс обучения. Особенность этих программ — малый размер — предопределяет результат.

«Будем учить работе с прикладными программами (tools)»

К концу промежуточного этапа нашей истории наблюдается совершенно иной подход. Аргументы в его пользу напоминают те, которые мы уже приводили в отношении компьютерной грамотности. Они звучат приблизительно так. Мощные прикладные программы сейчас широко используются в нашем обществе. Учащиеся почти наверняка столкнутся с ними после окончания школы или университета. Поэтому работать с ними нужно учить уже сегодня.

Необходимо отметить, что почти все прикладные средства — системы подготовки текстов, электронные таблицы, системы управления базами данных, графические программы — пришли из делового мира. Ни одно из них не было разработано для использования в учебном процессе. Я иногда называю этот этап компьютеризации образования собиранием крошек с конторского стола.

а) Деловые программы в классе

Эти средства создавались на основе представлений о потребностях делового человека. Для целей обучения они подходят не лучшим образом, и, похоже, это ускользает от многих их сторонников. Часто используется не совсем то, что разработано для деловых приложений,— упрощенные варианты, имитации для менее мощных компьютеров. В школах редко используются совершенные деловые программы, даже если имеется соответствующая техника.

Для обучения работе с текстовым редактором и другими прикладными программами часто возрождают курс компьютерной грамотности. Эти средства используют и при обучении различным предметам, и в некоторых случаях с пользой. Например, электронные таблицы могут применяться при обучении математике и естественным наукам. В литературе встречается много подобных примеров.

Следует особо выделить роль систем подготовки текстов в обучении письму. В дальнейшем текстовые редакторы найдут применение практически везде. Это неудивительно: возможность легко вносить исправления без утомительного переписывания и перепечатки, автоматическая проверка написания убедительно демонстрируют их потенциал.

Однако система подготовки текстов — лишь одно из компьютерных средств, которые могут быть полезны в процессе письма. Более того, обучение письму связано с использованием целой методологии, лишь частью которой является редактор текстов. Ключевая идея эффективного использования компьютера для развития письменной речи состоит в том, что существуют средства, оказывающие поддержку на этапах, предшествующих созданию текста, и средства, помогающие преобразовать уже созданный текст. Однако в настоящее время эти средства редко, и то лишь фрагментарно, объединяются в рамках целостной системы обучения письму и на более поздних этапах — творческому письму.

б) Прикладные программы и содержание обучения

Роль и место прикладных программ в учебных планах — важный аспект, которым часто пренебрегают. Для достижения цели — усвоения содержания учебных предметов — одних прикладных программ недостаточно. Необходимо разнообразное программное обеспечение, предназначенное для совместного с ними использования. Как учащимся овладеть этими средствами? Сколько ценного учебного времени отводить на освоение механизма их работы? Как эти средства сочетаются друг с другом и с прочими аспектами учебного процесса? Как довести до учащихся, что важны не средства сами по себе, а учебный материал, который усваивается с их помощью?

Эти вопросы часто остаются без ответа,

так как указанные программы вырваны из контекста деловых приложений, а педагогические аспекты их использования мало разработаны. Проблема в чем-то сходна с упоминавшейся выше проблемой «голой» имитации: акцент не на обучении, не на помощи ученику, а на технологии.

Даже если мы сумеем ответить на поставленные вопросы и предусмотреть в учебных планах место для прикладных программ, останется множество разделов, где они не имеют важного значения или более серьезную роль играют другие, основанные на технологии виды учебных материалов. Представление, весьма странное, о том, что в обучении можно обойтись лишь средствами, разработанными для деловых приложений, отражает скудные возможности современного образования.

«Будем использовать сети»

Во многих странах, а также в рамках отдельных проектов в конце концов достигают этапа, на котором внимание фокусируется на использовании сетей, сначала локальных, затем дальних. Обычного персонального компьютера уже недостаточно, учащимся необходимы связи между компьютерами и те возможности, которые обеспечивает сеть. Нет нужды, чтобы пользователи работали с сетью все время, многие дальние сети предполагают лишь эпизодическую пересылку сообщений по телефонной линии, иногда без ведома учащихся.

Сначала использование сетей почти полностью определяется соображениями удобства и учета. Удобство обеспечивается тем, что больше нет необходимости работать с множеством флоппи-дисков, учет — благодаря возможности регистрировать то, что делает или не делает учащийся. Следующая стадия — использование электронной почты и электронной доски, часто слабо связанных с содержанием обучения.

До сих пор практически не разрабатываются учебные материалы, которые требуют применения сетей, отличного от простого использования электронной почты и доски. Хотя, казалось бы, открываются интересные педагогические возможности, которые могут быть реализованы только с помощью сетей, к ним обращаются редко. Поэтому сети в большинстве случаев становятся дорогим дополнением, не оказывающим существенного влияния на учебную ситуацию. Однако, как и на других этапах компьютеризации, здесь имеется потенциал, который может быть реализован впоследствии.

«Будем разрабатывать системы управления»

Приходит время, обычно в начале второго этапа, когда обсуждается возможность применения технологии для управления учебным процессом. Аргументы в пользу систем управления вполне разумны. Учителя тратят много времени на бумажную работу. Если система управления поможет сократить это время, она будет с интересом встречена учителями.

Простейшие системы управления просто заменяют классный журнал, обеспечивая учителю возможность фиксации оценок и продвижения в изучении предмета. Учитель, вместо того чтобы записывать результаты выполнения учениками заданий, вводит их в режиме диалога, желательно с помощью дружественной программы запросного типа. При этом компьютер может выполнить какую-то математическую обработку, например вычисляет среднюю оценку. Такие системы представляют минимальный интерес.

Более совершенные системы управления могут быть непосредственно связаны с педагогическими программными средствами, обеспечивая автоматическую запись информации. Системы такого типа существуют, и можно ожидать увеличения их числа на завершающих этапах компьютеризации обучения.

Назначение систем управления не стало предметом широкого обсуждения. Часто считают, что фиксация результатов и подготовка различных отчетов нужны лишь учителю. Однако не менее, а в перспективе, возможно, и более, важны системы управления, непосредственно ориентированные на помощь учащемуся. Такие системы необходимы, если мы хотим, чтобы ученик понимал, над чем ему работать дальше. Следовательно, системы управления должны иметь несколько интерфейсов, предназначенных для различных пользователей.

Заключительный этап

До сих пор в описании событий мы стремились к исторической точности (хотя, как и в любой попытке охватить некоторое направление развития в целом, могут ускользать специфические особенности). Сегодня мы вынуждены отметить, что компьютер все еще не играет заметной роли в совершенствовании системы образования. Этап эффективного использования компьютера в классе еще не наступил ни в одной стране и ни в одной организации, хотя некоторые ближе к нему, чем другие. Однако наша история не закончена. Предстоит следующий — завер-

шающий — этап, на котором я коротко остановлюсь (в моих статьях и книгах заинтересованный читатель может найти более детальное описание).

Девиз завершающего этапа можно сформулировать так: «Будем разрабатывать целостные компьютерные учебные курсы». Потребуются весь накопленный опыт и осознание того, что единственный путь эффективного использования компьютера состоит в разработке совершенно новых курсов, с самого начала ориентированных на применение всех учебных сред, включая новейшие, интерактивной технологии. Эти курсы могут сильно отличаться от существующих по форме, содержанию и роли преподавателя. Их содержание изменится потому, что меняется наше общество, а структура — благодаря новым технологиям обучения.

Идея эффективного использования технологии в рамках законченных учебных курсов не нова. Можно вспомнить, например, курсы теории множеств, логики и языка, созданные в Стэнфорде, курсы физики и математики, созданные в Ирвине, и недавно разработанный курс «Письмо ради чтения», ориентированный на детей младшего возраста. Однако на разработку целостных курсов, основанных на интерактивной технологии, выделяются очень небольшие ресурсы. Большая часть средств по-прежнему вкладывается в описанные выше ранние этапы. Созданы сотни целостных курсов, основанных на видео, в том числе с элементами интерактивности, и лишь единицы целостных компьютерных курсов. А ведь без них нельзя оценить весь педагогический потенциал технологии.

Итак, основа завершающего этапа — создание целостных курсов, которые реализуют новые возможности и приведут к перестройке обучения в школах и университетах, неосуществимой в рамках старых технологий. В некоторых странах появляются черты такого будущего, но в целом мы к нему еще не готовы.

Целостные курсы должны сопровождаться всесторонним оцениванием. Важность проверки, как с созданными материалами работают учащиеся, уже подчеркивалась ранее. Желательно также, чтобы в каждой области было разработано несколько курсов, это позволит оценить их эффективность не только в сравнении с традиционными курсами, но и друг с другом.

Система образования испытывает огромные трудности, и справиться с ними можно, лишь заново перестраивая учебные курсы и школьное обучение в целом. Только таким образом можно приблизиться к завершающему этапу нашей истории, потенциал которого поразителен.

Перевод подготовлен
В. Н. КАПТЕЛИНИНЫМ

Компьютер для университетов

Компания Next (США), созданная С. Джобсом после ухода из Apple, предложила потребителям новый компьютер. В настоящее время возможности ПК линий IBM PC и Macintosh наращиваются только количественно, не переходя в новое качество, поэтому ПК «Next» (он назван так же, как и компания) представляет большой интерес: он может оказаться первой ласточкой нового витка развития вычислительной техники.

Основа «Next» — кубический корпус со стороной около 30 см, в котором размещены источник питания, системная плата и накопитель на стираемых оптических дисках. В корпусе есть гнезда еще для трех плат и двух НГМД для 5,25-дюймовых дисков. Источник питания мощностью 200 Вт автоматически настраивается на параметры электрической сети — ее напряжение может меняться от 90 до 270 В, частота — от 50 до 60 Гц.

Большая часть ИС, установленных на системной плате размером 280×280 мм, спроектирована фирмой Next, однако процессор — 68030 фирмы Motorola. Эта же фирма изготовляет используемые в ПК сопроцессор для операций с плавающей запятой 68882 и цифровой процессор сигналов 56001. Последний предназначен для обработки музыкальных и речевых сигналов, матричных вычислений и др. ИС 68030 и 68882 работают с тактовой частотой 25 МГц, средняя производительность центрального процессора — 12,5 млн. операций в секунду.

Объем размещенного на системной плате ОЗУ — 16М байт. По мере налаживания промышленного выпуска микросхем более высокой степени интеграции объем ОЗУ можно будет расширять до 64М байт. ВидеоОЗУ оформлено в виде отдельной секции на 256К байт.

Большой набор портов позволяет подключить «Next» к чему угодно без всяких сложностей. На плате есть видеопорт для дисплея «MegaPixel», порт для лазерного принтера, порт интерфейса SCSI (малых вычис-

лительных систем), два последовательных порта RS-422, специальный порт для непосредственной связи с центральным процессором и соединитель для подключения к сети Ethernet. Прямой доступ к памяти, реализованный с помощью 12 процессоров ввода-вывода, размещенных в одной микросхеме, позволяет не занимать время центрального процессора на обмены между ОЗУ и периферийными устройствами.

Дисплей нового ПК, названный «MegaPixel», имеет разрешающую способность 1120×832 точки, экран 43 см по диагонали, частоту регенерации изображения 68 кГц (это обеспечивает безупречное качество «картинки») и способен отображать четыре градации серого. Цветное изображение владелец «Next», увы, пока не может использовать, так как другие дисплеи не подходят к этому ПК. Впрочем, фирма обещает в будущем выпускать и средства цветной видеографики.

Дисплей подключается к системной плате кабелем, по которому подается питание и поступают сигналы. Клавиатура (имеющая 84 клавиши, общий для всех устройств «Next» выключатель питания и регуляторы яркости, контрастности и громкости) подключается к дисплею. В комплект может входить двухкнопочная «мышь». Подключение других «настольных» устройств не планируется.

Особенно интересна в «Next» внешняя память. Магнитооптическое ЗУ обеспечивает запись на диск 256М байт информации и скорость передачи данных около 1М байт в секунду. Отсутствие магнитных дисков — следствие желания разработчиков позволить каждому студенту носить с собой весь свой «компьютерный мир» — диск с полным набором программ и файлов, в том числе и весьма длинных, содержащих большие объемы текстовой, графической информации с высоким разрешением, цифровые записи музыки и речи. Гибких магнитных дисков для этого понадобилось бы слишком много, жесткие же требуют слишком аккуратного обращения. Так что оптические диски, возможно, окажутся оптимальным решением проблемы.

Другая интересная особенность — активная аппаратная поддержка возможностей звукозаписи и звуковоспроизведения. Процессор 56001 позволяет, в частности, осуществлять синтез мелодий в реальном времени. Стереофонический вывод обеспечивает качество звуковоспроизведения, аналогичное компакт-дискам; звуковой ввод имеет качество телефонного канала.

Лазерный принтер (за дополнительную плату в 2000 долларов) обеспечивает плотность печати до 16 точек на миллиметр. Это в точном смысле слова печатающее устройство: вся необходимая обработка данных и синтез печатаемого изображения производятся на системной плате «Next», принтер полностью управляется компьютером; на нем нет никаких переключателей — только разъем для кабеля подключения к ПК и шнур питания.

В цену ПК «Next» входит и программное обеспечение. Впрочем, пора, видимо, использовать новый термин — информационное обеспечение. Ведь это не только редактор текстов, система поддержки электронной почты и другие программы, но и, к примеру, записанное на диск полное собрание сочинений Шекспира. Главное же здесь, конечно, операционная система. Разработанная в Университете Карнеги-Меллона (США), ОС Mach выполняет три основных функции: обслуживает виртуальную память, распределяет время центрального процессора между одновременно выполняющимися прикладными программами и организует межзадачные связи, благодаря которым программы могут обмениваться данными и результатами между собой. Кроме того, естественно, выполняются и стандартные функции обслуживания файловой системы, сетевых обменов и т. п.

Для создания программ на новом компьютере можно использовать три мощных вспомогательных средства. Первое — комплект прикладных программ: библиотека, содержащая множество инструментальных процедур — управления окнами, создания меню и т. п. Второе — компилятор языка программирования Си с объектно-ориентированными расширениями. В программы для этого компилятора можно включать созданные ранее процедуры, в частности, из комплекта прикладных программ. Третье средство — так называемый построитель интерфейса, позволяющий формировать интерфейс прикладных программ, оперируя необходимыми объектами (окнами, меню и т. п.) в их графической форме, что резко упрощает программирование.

Стоимость нового компьютера на фоне его достоинств кажется небольшой — 6500 долларов. Нужно учесть, однако, что пока он продается только партиями по заказам высших учебных заведений, так что названная цена является оптовой.

Школьной информатике быть

Прочитав новый базисный учебный план и варианты переходного плана, мы были очень удивлены и обеспокоены, не обнаружив предмета «Основы информатики и вычислительной техники» ни в союзном компоненте, ни в республиканском. Есть предмет «Математика и информатика», на который не увеличено количество часов.

Похоже, что провозглашенные в 1985 г. компьютеризация и формирование компьютерной грамотности сворачиваются. А ведь сейчас преодолены многие трудности первых лет: появились различные учебники и учебная литература, подготовлено некоторое программное обеспечение, в школах есть учителя, которые могли бы и хотят вести этот предмет.

Мы считаем, что в школе должен быть предмет «Информатика и вычислительная техника» с соответствующей программой, отвечающей цели — обеспечение всеобщей компьютерной грамотности.

В нем необходимо познакомить учеников с новыми информационными технологиями и научить их использовать компьютер не только как вычислительное устройство, но и как средство быстрой обработки больших объемов информации. Школьники должны научиться обрабатывать тексты с помощью ЭВМ, производить поиск информации с помощью систем управления базами данных (СУБД), использовать графические и музыкальные редакторы в своей практической или творческой деятельности, использовать электронные таблицы для ведения бюджетных расходов, пользоваться сетями ЭВМ для передачи информации, иметь представление о том, как используется вычислительная техника в управлении различными технологическими, промышленными и другими динамическими процессами. Чтобы этому научиться, необходимо в разумных объемах знать структуру ЭВМ, операционные системы, уметь использовать внешние (периферийные) устройства, ориентироваться в многообразии программных средств.

В настоящее время в полном объеме реализовать такой подход затруднительно, потому что имеющаяся в большинстве школ техника просто не позволяет этого сделать. Кроме того, необходимо отказаться от порочной практики преподавания вместо информатики второсортного программирования в школах, оснащенных ВТ. К сожалению, многие этого пока не осознали. Вести курс программирования надо на факультативах и в кружках с ребятами, которые хотят этим заниматься и имеют способности.

Предмет «ОИВТ» необходимо преподавать и в тех школах, где пока нет ВТ. Ведь о том, что такое информатика, информация, как она может быть представлена, как она хранится, передается, обрабатывается, можно говорить с учащимися и без ВТ.

Преподаватели не только ОИВТ, но и других предметов отмечают, что в старших классах многие учащиеся не умеют логически мыслить. Поэтому изучение алгоритмизации целесообразно начинать с V класса. Этот курс должен включать обучение составлению алгоритмов, касающихся разнообразных сфер деятельности человека, развитие навыков алгоритмического мышления, логики, воспитание культуры системного мышления.

Чтобы курс ОИВТ был эффективен, преподавался на должном уровне, необходимы соответствующие программы. Для этого нужно проводить конкурс программ (машинный и безмашинный варианты), используя творческий потенциал учителей, методистов, программистов, научных работников, наконец, просто энтузиастов.

Итак, мы утверждаем: школьной информатике быть! Нужно сделать для этого все возможное.

О. К. ВАСИЛЬЕВА,
учитель ОИВТ СШ № 8 ст. Новосибирск,
В. А. ЗАВЬЯЛОВ,
учитель ОИВТ СШ № 128 ст. Инская,
Г. Г. ЗАКОВРЯЖИНА,
методист дорожного
методического кабинета
Западно-Сибирской ж. д.

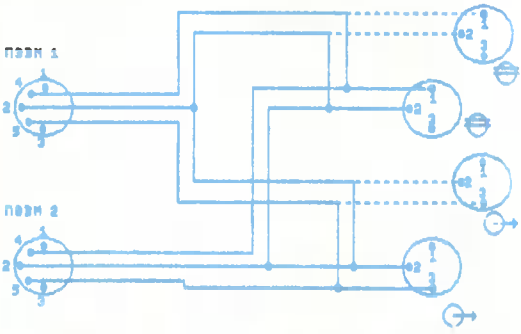
«Сура ПК 8000» в школе

1+1=... 3. Эта необычная формула нашла практическое подтверждение в Первомайском УПК Владивостока. Соединение двух шнуров обмена компьютера «Сура» с бытовым магнитофоном «Электроника» (схема перепайки шнуров приведена на рисунке) приводит к получению нескольких результатов:

для двух компьютеров вполне хватает одного магнитофона;

программу с кассеты можно загружать в оба компьютера одновременно;

можно осуществлять передачу программы с одного компьютера на другой.



Например, для перенесения программы УДАЧА с одного компьютера «Сура» на другой надо:

набрать на одном компьютере CSAVE «УДАЧА», а на другом CLOAD «УДАЧА» включить магнитофон на запись и нажать у обоих компьютеров клавишу ввода (возврат каретки).

В результате программа будет передана с машины на машину и записана на кассету. Если записывать программу не нужно, то у магнитофона следует нажать клавишу «пауза».

Доработку по соединению двух шнуров может сделать любой школьник, разбирающийся в радиотехнике. В результате экономятся средства при оборудовании компьютерного класса (магнитофонов требуется вдвое меньше). Преподавателю при загрузке программ требуется вдвое меньше времени. Учащиеся имеют возможность обмениваться программами. Автор этой рационализации — ученик XI класса СШ № 50 Сергей Осипов.

А теперь о самой машине. К сожалению, достоинств у нее меньше, чем недостатков.

В ПК «Сура» неплохая версия языка программирования Бейсик (если не считать, что многие нужные операторы языка в доработке у завода-изготовителя, а именно DRAW, LOC, CIRCLE, PAINT, PLAY, SWAP, TIME, OCT ¤ и некоторые другие). Хорошее и продуманное расположение клавиш.

В скудном руководстве по эксплуатации перечислены дополнительные узлы и блоки, которые можно подключить к «Суре», в том числе и печатающее устройство. А в примечании к руководству сказано, что данные узлы и блоки в комплект поставки не входят. Мы приобрели печатающее устройство СМ-6337. Это хороший принтер для машин «Искра 1030» и «ЕС 1840», его можно подключить к другим отечественным персональным компьютерам (так написано в руководстве по эксплуатации печатающего устройства).

Однако в руководстве отсутствуют какие-либо схемы компьютера. Кабеля для соединения компьютера с принтером нет ни в комплекте ПЭВМ, ни в комплекте принтера. Нет даже штеккера для «Суры».

Так что Пензенский завод ВЭМ, выпускающий компьютеры «Сура», с одной стороны, нахваливает машину, сообщает в руководстве о возможностях ее расширения с помощью дополнительных блоков и узлов, а, с другой — «обрубает» всякую возможность это сделать ввиду отсутствия схемы машины и необходимых для этого запчастей.

Профилактическое обслуживание и ремонт необходимо производить с любой ПЭВМ. Так почему в руководстве по эксплуатации нет никакой информации по этому вопросу, не указаны нормы на расходные материалы, требующиеся при профилактике (бязь, кисточки, спирт и т. д.)?

А тот факт, что программное обеспечение на кассете приходит с бракованной записью, а иногда эта запись вообще отсутствует!

Хотелось бы узнать мнение других специалистов об этой вычислительной машине. Мы также разработали игровые программы для «Суры». Заинтересовавшиеся могут обращаться по адресу: 690031, Владивосток, ул. Вязовая, 1, УПК Первомайского РОНО.

Т. МИТРЯШОВА

Доска объявлений

Прошу откликнуться владельцев МВЦ «Партнер-01.01», знающих организации, занимающиеся распространением программ для этого ПК.

Сообщаю, что резидентный интерпретатор Бейсика выполняет некоторые директивы, не указанные в руководстве: AT, CLS, CUR (задание координат на экране), HOME, LINE, PLOT, PAUSE (приостанов программы, при аргументе 200 длительность паузы 1 с), AUTO.

А. В. ИВАНОВ.
683056 Петропавловск-Камчатский, ул. Крашенинникова, 38, кв. 15.

Хочу переписываться с владельцами «Партнера-01.01» и «Кристы».

А. МАСЛЕННИКОВ.
129090 Москва, Колхозная пл., 1, кв. 101.

Обращаюсь к работающим на УКНЦ и имеющим хоть какое-нибудь программное обеспечение.

Е. М. ИВАНОВА.
446552 Куйбышевская обл., Сергиевский р-н, п. Суходол, ул. Парковая, 17—5.

Готов безвозмездно помочь начинающим пользователям УКНЦ. Имею компиляторы Фортрана, Паскаля, Си, обучающие и игровые программы.

В. В. ПЕРУШЕВ.
107078 Москва, ул. Садово-Черногрязская, 5/9, кв. 85.

Хочу переписываться и обмениваться программами с владельцами «Радио-86РК».

А. Н. ДИТЦ.
624488 Волчанск, ул. Краснотурьинская, 15—15.

Имею более 200 системных и игровых программ для «Микроши» и «Кристы». Готов поменяться и поделиться. Высылаю каталог. Интересуюсь программами на Форте и Паскале.

В. В. МОКРЯК.
663310 Норильск, ул. Талнахская, 1, кв. 128.

Ищу программы для ПЭВМ «Правец-8Д».

А. Н. ГАРАНЬ.
340044 Донецк, ул. Б. Магистральная, 19, кв. 4.

Хочу обмениваться программами для «Ага-та», имею около 30 программ.

В. В. ГОФШТАДТ.
703050 Самарканд, ул. Октябрьская, 23-а, кв. 8.

Владельцев ПЭВМ «Вектор-06Ц» приглашаю обменяться программами и т. п.

В. КАРПОВ.
603005 Горький, ул. Пискунова, 3, корп. 3, кв. 147.

Хотел бы переписываться с владельцами «Вектора-06Ц», проживающими в городах, где этот компьютер производят: Кишеневе, Минске, Астрахани, Волжском. С радостью готов обмениваться программами.

Д. Ю. БЕЛЯЕВ.
610002 Киров, ул. Воровского, 17, кв. 4.

Ищу возможность приобретения программ для «Вектора-06Ц», партнеров для обмена программами.

В. Е. СМИРНОВ.
400137, Волгоград, бульв. Победы, 66, кв. 16.

Хочу переписываться с обладателями компьютеров «Атари» серий XE/XL. Поменяюсь программами, схемой подключения к цветному телевизору без ПАЛ-блока, интерфейсом Турбо-2000.

Т. В. ОВЧИННИКОВ.
202500 Валга, а/я 162.

Прошу откликнуться пользователей ПК «Атари». Имею большой фонд программ, литературы и т. п. для этого компьютера.

С. В. ВЕНСКО.
222810 Минская обл., Пуховичский р-н, г. Марьина Горка — 4, дом 20, кв. 72.

Предлагаю для обмена большой набор программ для «Атари XE», «Коммодор 64».

А. П. КАЗАНОВ.
210026 Витебск, а/я 61.

Приглашаю к сотрудничеству, обмену программами всех пользователей МК-61 и МК-52.

М. Т. ЛЯЛИН.
117465 Москва, ул. Тюленева, 41, кв. 108.

Калининградский научно-технический кооператив «ИНФО-сервис»

предлагает пользователям КУВТ-86 (Бейсик) и ПЭВМ «Агат» новый набор прикладных программ специально для целей обучения

Преподавание информатики невозможно без изучения основных типовых программ, таких, как системы управления базами данных, различные редакторы и числовые таблицы.

Учитывая, что большинство таких программ, написанных энтузиастами программирования, неудобны для использования на уроке, с 1987 г. в Учебно-компьютерном центре Ленинградского роно Калининграда разрабатывался пакет программ специально для целей обучения, сочетающий простоту и наглядность, заботу о неподготовленном пользователе и надежность в работе.

Программы пакета легко осваиваются учениками и содержат:

элементы программирования в машинных кодах;
защиту от ввода неверных по смыслу данных;
сопроводительную документацию, примеры и методику использования.

1. Пакет ПП «КЛАСС» для КУВТ-86 (Бейсик)

СИСТЕМА ТЕСТОВОГО КОНТРОЛЯ (см.: ИНФО. 1989. № 1). Неквалифицированный пользователь (учитель-предметник), имея на экране специально выделенные поля, готовит в диалоговом режиме задания и ответы. Из них на ДВК формируется программа-тест для ученика. Результаты собираются в отдельном файле (или на LP:) и могут вноситься в «ЖУРНАЛ», созданный на базе СУБД на ДВК. БАЗА ДАННЫХ для БК-0010 совмещает в одной программе функции поиска и редактирования, хранит до 100 записей, использует диск или магнитофон, выбирает информацию через цепочки условий типа И/ИЛИ, исключительно проста в управлении.

ГРАФИЧЕСКИЙ РЕДАКТОР для БК-0010 рисует графическим курсором по восьми направлениям, строит цветное изображение типовыми формами, выбирает обычное или ускоренное движение курсора, координаты и цвет которого выводятся в служебной строке. Созданное изображение записывается на диск или магнитофон и может в дальнейшем редактироваться или выводиться на печать.

ТЕКСТОВЫЙ РЕДАКТОР для БК-0010 имеет текстовое поле в шесть экранных страниц и полный набор печатаемых символов. Текст сохраняется на диске, магнитофоне и может быть распечатан непосредственно.

ГЕНЕРАТОР-РЕДАКТОР МУЗЫКАЛЬНЫХ ПРОГРАММ БК-0010 имитирует клавиатуру музыкального инструмента (до трех октав), генерирует и записывает на диск или магнитофон программу, воспроизводящую набранную мелодию; последняя может быть отредактирована.

2. ПЭВМ «Агат»

ИНФОРМАЦИОННО-ПОИСКОВАЯ СИСТЕМА с древовидной организацией данных, работающая в режиме меню, написана на языке Ассемблера и имеет собственный формат записи данных, независимых от системы; до семи томов (дисков) для одной базы данных; до пяти уровней организации данных; до 32 разветвлений по каждому узлу дерева на каждом уровне. Это дает до 32 554 432 оконечных информационных полей.

МНОГОЦЕЛЕВОЙ РЕДАКТОР ТЕКСТОВ И ОБРАЗОВ написан на языке Ассемблера и является мощным средством создания библиотек образов (шрифты, картинки) и печати созданных с их помощью изображений и/или текстов. Включает оригинальные функции и масштабирование.

Указанные программы являются полностью оригинальными и прошли испытания в период учебных занятий с 1988 по 1990 г.

Наш адрес: 236016, г. Калининград, а/я 878. КНТК «ИНФО-сервис».

Проект «Пилотные школы»

Фирма IBM традиционно уделяет большое внимание сфере образования. С началом выпуска и распространения в средних учебных заведениях персональных компьютеров интерес фирмы к этой сфере особенно возрос. В частности, в 1983 г. фирма приступила к осуществлению программы компьютеризации образования (Secondary School Computer Education Program), на которую было выделено 8 млн. долларов и в которой приняли участие 12 институтов переподготовки учителей и 89 школ (high school).

С появлением нового семейства компьютеров — IBM PS/2 в ряде штатов США развернута целая сеть так называемых пилотных школ (PILOT* SCHOOLS).

Свою деятельность в области образования фирма не ограничивает пределами США. В Европе, например, эта работа проводится через сеть представительств. Базовыми являются международные учебные центры в Бельгии, Швеции, Испании, Франции.

Свидетельством интереса фирмы к проблемам информатизации образования в СССР явился симпозиум, проведенный совместно с Гособразованием СССР в сентябре 1989 г. в Москве. По предложению фирмы было принято решение осуществить в течение 1990—1991 гг. три совместных проекта: «Дети-инвалиды», «Пилотные школы», «Обучение менеджеров». Фирма взяла на себя поставку и сопровождение техники и базового программного обеспечения, а также обучение советских специалистов, принимающих участие в проектах.

Проект «Пилотные школы» включает следующие этапы:

подготовку участников проекта;

разработку программно-методических комплексов для проведения психолого-педагогического эксперимента;

обучение учителей Пилотных школ;

проведение эксперимента в Пилотных школах в 1990—1991 гг. и в 1991—1992 гг.;

обработку и анализ результатов;

выработку рекомендаций по формам и методам использования НИТ фирмы IBM в отечественной средней школе.

В проекте «Пилотные школы» участвуют две средние школы Москвы: № 1217, № 1234. В каждую из них поставлен комплект вычислительной техники:

PS/2 модель 30 (процессор INTEL 8086, HCGA) (FD1.44 MB, HD30 MB) — 10 шт.

PS/2 модель 50 (процессор INTEL 80286, VGA) (FD1.44 MB, HD60 MB) — 1 шт.

Комплекс аппаратных средств поддержки локальной сети — Token ring (быстродействие 4 Мбод).

Принтер — 1 шт.

Комплекс программного обеспечения включает:

MS DOS версии 3.3 и 4.0;

программное обеспечение поддержки локальной сети — LAN;

пакет, ориентированный на деловое применение,— WISEPACK ASSISTANT SERIES;

графический редактор, позволяющий программировать взаимодействие пользователя с последовательностью графических кадров,— STORY BOARD PLUS;

инструментальное средство для подготовки обучающих программ — LUNKWAY; BASIC; PASCAL.

Для подготовки программных и методических материалов, а также координации психолого-педагогического эксперимента в Пилотных школах, обработки и анализа результатов эксперимента создан учебно-научный методический центр новых информационных технологий (УНМЦ НИТ).

Центр оснащен шестью ПЭВМ IMB PS-2

* От англ. pilot — опытный, экспериментальный.— *Примеч. ред.*

модель 50. В нем работают 20 методистов и 6 программистов. Центр создан на базе акционерного социалистического предприятия «Компьютерный учебно-демонстрационный издательский центр (КУДИЦ)», среди акционеров которого — Гособразование СССР, Союзвузприбор, АН СССР. Перед его сотрудниками стоят две основные задачи:

разработка новых форм и методов использования НИТ в обучении;

разработка методик переподготовки учителей.

Подготовка учителей Пилотных школ

Курс ориентирован на слушателей, не имеющих специальной подготовки в области вычислительной техники и программирования.

Основной целью курса является подготовка учителей Пилотных школ к проведению уроков и внеклассной работы с использованием компьютеров и программных средств, предоставленных фирмой IBM. Курс должен также заложить основу для активного участия учителей Пилотных школ в разработках программных и методических материалов, проводимых УНМЦ НИТ.

Общий объем курса составляет 120 ч (см. таблицу). В результате изучения курса слушатели должны знать:

основные сведения об устройстве, принципах работы и использования персональных ЭВМ типа IBM PS/2;

состав и назначение основных комплексов программного обеспечения, передаваемых в Пилотные школы;

основные формы и методы использования компьютеров в учебном процессе и внеклассной работе со школьниками, а также формы и методы использования ПЭВМ при решении некоторых административных задач в рамках средней школы.

Слушатели должны уметь:

выполнять все операции с IBM PS/2, необходимые для подготовки и проведения уроков и внеклассной работы;

использовать текстовый процессор для подготовки и печати несложных документов;

использовать электронные таблицы для выполнения расчетов;

осуществлять операции поиска и коррекции информации в базе данных с простой структурой;

представлять числовую отчетную информацию в виде диаграмм;

готовить иллюстрационный материал с помощью графического редактора;

использовать для подготовки фрагментов уроков и проведения занятий инструментальную систему подготовки учебных курсов.

Содержание курса

Наименование раздела	Общий объем (ч)	Лекции (ч)	Практическая работа (ч)
Основы применения ПЭВМ в обучении	16	8	8
Основы ЭВМ. Введение в ОС	24	9	15
Деловые применения ПЭВМ	32	7	25
Знакомство с компьютерной графикой	16	0,5	15,5
Система подготовки автоматизированных учебных курсов Linkway	32	10	22
Всего	120	34,5	85,5

Материал подготовил Б. КИСЕЛЕВ,
научный руководитель УНМЦ НИТ, Москва

Банк педагогических данных МОПИ им. Н. К. Крупской

В ряду проблем педагогической науки, не получивших до настоящего времени достаточного разрешения, несмотря на пристальное внимание к ней, стоит проблема обобщения и распространения педагогического опыта. Опыт работы учителей и других категорий работников народного образования аккумулирует в себе поиски и находки педагогической практики, позитивные и негативные тенденции ее развития и является одним из важнейших источников научно-педагогического знания. Вместе с тем обобщенный педагогический опыт отражает те изменения в практике, которые происходят под влиянием педагогической науки, и в этом смысле может служить критерием ее эффективности. К сожалению, существующие сегодня способы обобщения информации об опыте, опирающиеся на текстовое его описание, не обладают достаточной полнотой и оперативностью и не позволяют обеспечить информационный кругооборот между педагогической наукой и практикой. Сложившееся положение приводит к тому, что учитель, в лучшем случае, может опираться лишь на опыт немногих своих коллег, более или менее полно описанный в специальной литературе или предметно-методических журналах. С другой стороны, педагогическая наука, не располагая достаточной информацией об опыте работы школы, зачастую ставит и решает исследовательские задачи, не являющиеся актуальными с точки зрения создания научного обеспечения поступательного движения педагогической практики.

Развитие различных отраслей знаний в течение последних десятилетий показало, что надежный кругооборот информации в системе «наука — практика» может быть обеспечен на базе автоматизированных информационных систем с использованием вычислительной техники. Такие системы, не подменяя ни в коей мере печатной продукции, могут обеспечить оперативный сбор, обобщение, классификацию, хранение и распространение информации.

Исследования, выполненные на кафедре педагогики МОПИ им. Н. К. Крупской под руководством проф. В. И. Журавлева, позволили создать новую технологию машинно-ориентированного отражения и распространения информации о педагогическом опыте. В ее основу положены представления о том, что педагогический опыт может быть описан как более или менее выраженная система, включающая в себя методы, средства, алгоритмы действий, правила, приемы, формы педагогической деятельности. В качестве системообразующих компонентов опыта могут также выступать задачи и содержание обучения и воспитания, показатели и критерии эффективности педагогического процесса, условия его проведения и другие виды педагогического творчества, отражаемые в категориях педагогической науки. Отдельные находки педагогов-практиков, не образующие целостной системы, также могут быть отнесены к одной из названных категорий.

Возможность структурирования информации об опыте позволяет в качестве единицы хранения и использования ее в АИС выбрать информационно-педагогический модуль (ИПМ).

ИПМ — краткое конструктивное описание информации, удовлетворяющее следующим требованиям:

логическая законченность (информация, записанная в виде ИМП, должна носить целостный характер и описывать конкретную часть опыта, отнесенную к одной из педагогических категорий);

информативность (информация, представленная в ИПМ, должна быть достаточной для ее использования в практике без обращения к дополнительным источникам информации);

популярность (язык записи ИПМ должен быть доступен педагогу любой квалификации).

Изложение педагогической информации в форме ИПМ позволяет ускорить процесс ее усвоения более чем в 13 раз по сравнению с традиционным способом (книга, статья, реферат и др.) без потери конструктивного смысла перерабатываемого материала.

В настоящее время в МОПИ им. Н. К. Крупской на базе описанных подходов создан банк педагогических данных (БПД), отличительными чертами которого являются:

использование в качестве технической базы персональных компьютеров;

непосредственный диалог пользователя любой квалификации с машиной;

возможность быстрого развертывания филиалов банка в любых заинтересованных организациях, включая школу.

В БПД накапливается информация о педагогическом опыте по различным отраслям обучения и воспитания.

Редакция журнала «Информатика и образование» и МОПИ им. Н. К. Крупской приглашают учителей информатики и других педагогов, использующих вычислительную технику в обучении и воспитании, принять участие в накоплении информационного фонда БПД. Материалы о своих педагогических находках высылайте по адресу редакции нашего журнала.

Квалифицированные специалисты МОПИ им. Н. К. Крупской осуществят экспертную оценку вашего опыта и предмашинную обработку информации для включения ее в информационный фонд банка. Наиболее интересная информация будет публиковаться на страницах журнала. Защита авторских прав гарантируется. Вкладчикам банка предоставляется приоритетное право пользования полным объемом его информационного фонда.

Ждем ваших сообщений!

А. ЕЛИЗАРОВ,
зав. лабораторией БПД,
кафедра педагогики

Адрес для справок: 107005, Москва, ул. Радио, 10А, МОПИ им. Н. К. Крупской

Уважаемые читатели!

Подписная кампания этого года омрачена резким повышением цен на большинство периодических изданий. Многие из вас, оформляя подписку, с недоумением обнаружили, что сия горькая чаша не минула и нас.

Что же произошло? По выступлениям центральной печати вы уже знаете, что с 1991 г. практически все периодические издания будут работать в жестких экономических условиях: резко (на сотни процентов) возрастет стоимость бумаги, услуг Минсвязи СССР по доставке печатных изданий и стоимость полиграфических работ.

При сохранении номинала (стоимости одного номера) суммы доходов от реализации тиража «ИНФО» не хватит даже на покрытие перечисленных расходов.

Чтобы сохранить журнал, который подобно большинству специализированных изданий и раньше не мог похвастать сверхприбылями — удавалось разве что сводить концы с концами, мы были вынуждены пойти на увеличение номинала.

И все же очень надеемся, что число наших подписчиков не сократится. Ведь «ИНФО», как свидетельствует редакционная почта, необходим и несмотря на возросшую цену по-прежнему остается, пожалуй, самым дешевым компьютерным изданием.

Еще есть время подписаться!

Обращаем внимание учителей информатики:
школы имеют право на подписку по безналичному расчету в пределах отпущенных им сумм. Более того, в письме-обращении Гособразования СССР (№ 93—11—123/ИН—08 от 06.08.90) к органам народного образования, призывающем помочь школам в организации подписки на педагогические журналы на 1991 г., говорится: «... в 1990 г. повышены расчетные нормативы ассигнований на приобретение учебно-методической и художественной литературы по безналичному расчету... Эти ассигнования позволяют каждой школе полностью удовлетворить потребность в педагогических журналах...»

Не упустите эту возможность!

ЧТО МОЖЕТ ЭВМ?

«Пулемет» для деталей

Современная технология производства сверхбольших интегральных схем достигла такого уровня, что компьютер вместе с необходимой для размещения программ памятью, схемами для взаимодействия с внешними устройствами и другой начинкой может разместиться в одном чипе. И все же на предприятиях, производящих электронику, еще достаточно сборочных линий, на которых на печатную плату устанавливаются так называемые дискретные элементы: резисторы, конденсаторы, полупроводниковые приборы.

Чтобы избавить людей от монотонной, а главное, не допускающей ошибок работы, различные фирмы выпускают автоматические монтажные устройства. Зарядив такой автомат бумажной лентой, на которую в определенном порядке наклеены различные (или однотипные) монтажные элементы и запустив соответствующую программу, оператор может заняться другими электронными монтажниками. Если лента достаточно длинная (она может достигать 5 км!) и печатных плат в подающем бункере хватает, то можно выкроить время и для отдыха.

Японская фирма OKANO предложила рынку лучший в своем классе (по оценке фирмы) монтажный аппарат менеее чем за 40 тыс. долларов. Деньги немалые, но и аппарат хорош: высокоскоростной, позволяющий использовать до 20 рулонов напичканной деталями ленты одновременно. С фантастической точностью (0,03 мм) и скоростью (0,9 с на компонент) установки элементов на плату вряд ли сравнится и бригада сборщиков, а уж фанатичная исполнительность, определяемая гибким программным обеспечением, не может быть превзойдена никем. Если добавить к этому, что два таких агрегата можно объединить в тандем для выполнения комплексной работы, то японский технологический феномен станет чуть менее загадочным.

Кнопка-дисплей

Вводить информацию в компьютер можно не только с помощью клавиатуры, но и мышью, джойстиком, прикасаясь кончиком светового пера или обычной ручки к экрану монитора, и даже голосом. Выводится информация не менее разнообразно: на экран монитора, бумагу, с помощью синтезатора или в виде электрических сигналов для различных роботов. Совместить две эти функции попытались японские изобретатели, которые разработали клавиатуру, каждая клавиша которой представляет собой маленький, умещающийся под пальцем, экран монитора.

Пойти на это их заставил необычно богатый графический алфавит японского языка, не умещающийся ни на одной клавиатуре разумного размера. А с помощью клавиатуры, каждая кнопка которой является газовоплазменным дисплеем, позволяющим отобразить любой из 7 тыс. символов японского языка, можно, даже не владея искусством каллиграфии, вполне сносно написать любое сообщение.

UNIX в картинках

В отличие от популярной операционной системы MS DOS, доминирующей на персональных компьютерах фирмы IBM, ОС UNIX, хотя и более удобная, по мнению системных программистов, не получила столь широкого распространения. И дело, видимо, в том, что более мощная ОС UNIX намного строже своей конкурентки, предоставляющей пользователям массу удобств в виде турбосистем, меню, пиктограмм и прочих графических и информационных удобств.

И вот наконец и UNIX уступил требованиям рынка. Американская фирма Intergraph предлагает привередливым пользователям новую рабочую станцию на базе персонального компьютера Macintosh, снабженную мощной иконографической версией операционной системы UNIX. Система очень проста в использовании. Теперь нет нужды изучать синтаксис системы — достаточно овладеть системой меню и интерактивными окнами. А контекстно-чувствительная помощь и простые системные сообщения об ошибках избавят от необходимости изучать толстые руководства. Кроме того, система поддерживает редактирование и создание новых графических экранных образов. Это дает возможность каждому создавать именно ту обстановку, которая для него удобнее всего.

Очевидно, освоить систему в графическом варианте будет значительно проще, чем изучить абстрактный командный язык shell классической ОС UNIX. Теперь эта операционная система получит новую жизнь. Ведь она мощнее других, а теперь, будем надеяться, стала и удобнее.

Не устает, не забывает

Никогда не забудет в нужный момент отключить магнитофон и включить нужную радиопрограмму новая автомагнитола японской фирмы Clarion. Если, конечно, она запрограммирована на интересующую ее владельца передачу.

Встроенный микропроцессор отслеживает несколько регулярных ежедневных радиопрограмм, работает будильником и управляет работой всего радиокомбайна, освобождая внимание водителя для более важного дела — управления автомобилем.

СП Диалог

Всем, кто имеет персональный компьютер.
Производительность вашего труда зависит
от скорости работы с клавиатурой.
Освоить слепой десятипальцевый метод письма на

русском, английском и национальных языках

вам поможет тренажер клавиатуры

ТРК 3.0

ТРК будет полезен
операторам ЭВМ и программистам,
научным работникам и инженерам,
деловым людям и руководителям,
журналистам и писателям,
бухгалтерам и плановикам,
машинисткам и секретарям-делопроизводителям.

Начинающим ТРК позволит освоить слепой метод за несколько дней.
Тем, кто уже владеет методом, поможет улучшить свой стиль
и повысить скорость. Профессиональным машинисткам даст возможность
легко перейти с пишущей машинки на компьютер.

ТРК — это снижение утомляемости глаз и рук,
повышение эффективности использования компьютера.

Обучение ведется в увлекательной игровой форме.
Специально разработанная методика обучения
ориентирована на клавиатуру персонального компьютера.

ТРК автоматически анализирует ошибки,
стиль письма обучающегося и выдает соответствующие рекомендации.
ТРК предназначен для использования на персональных компьютерах
типа PC/XT/AT и любых совместимых моделях,
в том числе и отечественного производства.
Имеется возможность работы с гибкого диска.
Цена 455 руб.
При покупке сразу нескольких экземпляров предоставляется скидка.

Наш адрес: **119899, Москва, Ленинские горы,**
Центр СП ДИАЛОГ в МГУ, проект ТРК.
Телефон: 939-38-91. Телефакс: 939-38-93.

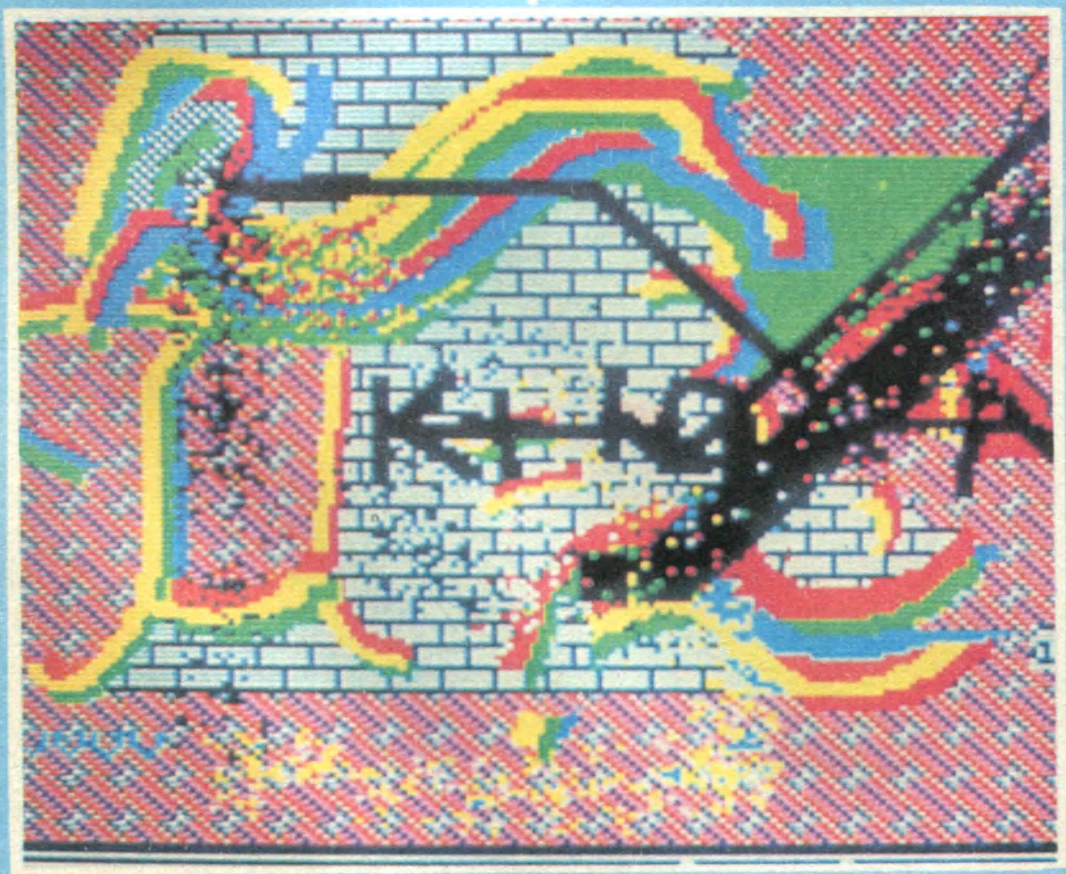
Юность 11-91

Цена 60 коп.
70423

ИНФО

5'90

ИНФОРМАТИКА И ОБРАЗОВАНИЕ



ISSN 0234—0453. Информатика и образование, 1990, № 5, 1—128.