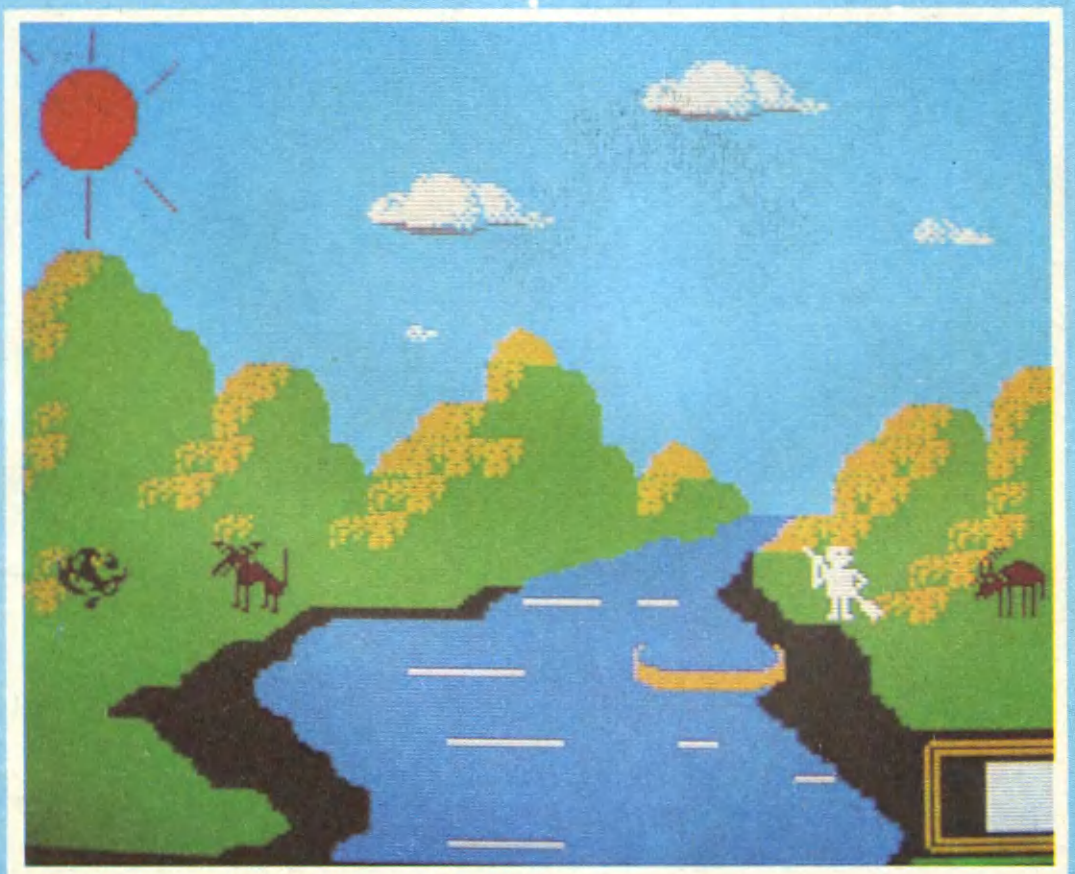
ISSN 0234—0453

ИНФОРМАТИКА И ОБРАЗОВАНИЕ

5 1989



Компьютеризация охватывает все более ранние ступени образования и воспитания. Сейчас требует ответа не столько вопрос «в каком возрасте начинать приобщение детей к компьютеру?», сколько «как это делать?».

В публикуемых в этом номере материалах рассказывается о подходе к преподаванию информатики младшим школьникам и о компьютерных комплексах для детских садов.

Научно-методический журнал
Государственного комитета СССР
по народному образованию

Издается с августа 1986 г.
Выходит шесть раз в год
Москва «Педагогика» 1989

ИНФ 5 '89

ИНФОРМАТИКА И ОБРАЗОВАНИЕ

сентябрь — октябрь

OldPC.su
7002
музей компьютеров

Содержание

Общие вопросы

Методика обучения

КВТ

Педагогический опыт

Обложка *Э. Бажилина*
В оформлении номера принимали участие: *Э. Бажилин, Н. Горбунова, С. Расторгуев*

Главный редактор академик
В. А. МЕЛЬНИКОВ

Редакционная коллегия
И. Н. АНТИПОВ
В. Н. АФАНАСЬЕВ
И. М. БОБКО
Г. В. ГОДЖЕЛЛО
С. А. ЖДАНОВ
Б. В. ЛОМОВ
Ю. В. ЛУИЗО
(зам. главного редактора)
Н. Г. МЕЛЬДИАНОВ
И. С. ОРЕШКОВ
О. К. ПАВЛОВА
А. Ю. УВАРОВ
А. И. ФУРСЕНКО
В. О. ХОРОШИЛОВ
К. В. ШЕХОВЦЕВ
(редактор отдела)

Редактор отдела *А. Кравцова*
Научный редактор *Т. Драгныш*
Зав. редакцией *Н. Игнатова*
Художественный редактор *Л. Розанова*
Корректоры *А. И. Сорнева, Л. В. Яковлева*

Сдано в набор 20.07.89. Подписано в печать 06.09.89. А 02018.
Формат 70×100/16. Бумага офсетная № 1. Печать офсетная. Усл. печ. л. 10,40.
Усл.-кр. отт. 42,88. Уч.-изд. л. 13,59. Тираж 83 395 экз. Заказ 1793. Цена 60 коп.

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

Адрес для переписки: 107005, Москва, Лефортовский пер., 8.
Адрес редакции: Студенческая ул, 37.
Телефон редакции: 249-97-77

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

© «Педагогика», «Информатика и образование», 1989

В настоящее время в различных странах, в том числе и у нас, пришли к выводу о невозможности дальнейшего развития систем образования только традиционными путями. Основным инструментом содержательной методологической и организационной перестройки образования становится процесс его информатизации, который несет в себе новые потенциальные возможности.

В предлагаемой вниманию читателей статье сделана попытка систематизировать имеющийся в нашей стране опыт в области реализации идей информатизации образования. Обрисованы лишь первоочередные проблемы, решение которых в большинстве случаев еще предстоит найти.

А. ШАТРОВ
Гособразование СССР, Главное учебно-методическое управление высшего образования
Ю. ЦЕВЕНКОВ
НИИ проблем ВШ

Проблемы информатизации образования

Важнейшая роль в развитии современного общества в последней четверти XX в. бесспорно принадлежит информатизации. По характеру и значению для будущего человеческой цивилизации процесс информатизации можно сравнить с переходом человечества от аграрного общества к индустриальному.

В результате внедрения новых информационных технологий (НИТ) значительно изменяются общественно-экономические структуры, формируется гибкое динамическое общество, способное к активной самооценке и выбору целей развития, быстрой и адекватной реакции на изменение внешней и внутренней среды.

Информатизация общества в нашей стране рассматривается как один из основных элементов государственной политики. Более 20 правительственных решений, вышедших в последнее время, в той или иной степени посвящены вопросам информатизации.

Под информатизацией общества понимается всеобщий социальный процесс производства и повсеместного использования информации как общественного ресурса, обеспечивающего интенсификацию экономики, ускорение научно-технического прогресса страны, процессов демократизации и интеллектуализации общества.

Информатизация предусматривает массовое внедрение методов и средств сбора, обработки, передачи и хранения информации на базе средств вычислительной техники (ВТ) и средств передачи информации.

Информатизация общества в нашей стране направлена на повышение уровня жизни и социальных потребностей советских людей, укрепление социальных и политических институтов, структурную перестройку народного хозяйства, интенсификацию развития экономики и ускорение научно-технического прогресса.

В процессе информатизации общества можно выделить пять основных направлений:

1. Комплексная автоматизация средств труда, технологических и производственных процессов.

2. Информатизация научных иссле-

дований, проектных работ и технологической подготовки производства.

3. Информатизация организационно-экономического управления.

4. Информатизация сферы услуг и быта населения.

5. Информатизация процессов обучения и подготовки кадров.

В информационном обществе центр тяжести в общественном разделении труда перемещается на производство, переработку и использование информации, резко возрастает наукоемкость общественного производства, повышаются требования к уровню общекультурной и общенаучной подготовки всех участников общественного производства.

Требование общества о необходимости создания условий для непрерывного образования его членов неразрывно связано с развитием информатизации образования. Непрерывное образование и информатизация являются фактически двумя сторонами одного и того же процесса — повышения компетентности членов общества.

Для обеспечения непрерывности образования, что также важно для его информатизации, необходимо в процессе обучения усилить фундаментальность подготовки и воспитать потребность учиться, привить уважение к знанию и его носителям.

Система непрерывного образования в условиях информатизации общества направлена на:

повышение уровня интеллектуальности общества за счет роста качества, интенсивности и персонализации обучения;

расширение возможности самообразования и переквалификации в системе неформального образования;

переход от информационно-репродуктивного типа учения на активно-творческий, продуктивный, в основу которого должна быть положена выработка потребностей и умений самостоятельного приобретения знаний, методов их пополнения и применения с использованием НИТ;

раннее выявление и адекватное дифференцированное обучение одаренных;

сокращение средних сроков обязательного обучения;

привитие всем категориям обучаемых практических навыков работы со средствами информатизации;

подготовку высококвалифицированных специалистов по информатике и ВТ.

На Всесоюзном съезде работников народного образования в декабре 1988 г. обсуждена и принята государственная концепция непрерывного образования.

Одна из сегодняшних основных задач — разработка концепции информатизации образования и соответствующей государственной программы.

Информатизация общества и связанное с ней широкое распространение ВТ, средств коммуникации, методов обработки информации и т. п. требуют совершенствования содержания подготовки, обновления состава учебных дисциплин, уточнения перечня специальностей, делают саму систему образования в определенной степени «объектом» информатизации.

В вопросах совершенствования содержания подготовки следует отметить в первую очередь введение в учебный процесс дисциплин, направленных на общеобразовательную и профессиональную подготовку в области информатики.

Информатика должна рассматриваться как учебная дисциплина, которая наряду с математикой, общественными науками, родным и иностранным языком обеспечивает формирование интеллектуальных способностей у обучаемых, способствует демократизации и гуманизации процесса образования в целом. Информатика в учебном процессе может быть как самостоятельной, так и прикладной дисциплиной.

Началом практического решения проблем информатизации образования в масштабе страны следует считать 1985 г., когда было принято правительственное решение об обеспечении компьютерной грамотности учащихся и создании условий для широкого использования средств ВТ в учебном процессе.

В настоящее время в образовании складывается единая система непрерывной подготовки в области информатики, включающая базовую (общеобразовательную) подготовку и специальную, расширяющую знания обучаемо-

го по различным направлениям прикладной информатики. Система непрерывной подготовки охватывает все уровни образования, включая дошкольное воспитание.

В ряде детских садов проводится эксперимент по использованию компьютерных игровых комплексов, разработанных под руководством НИИ дошкольного воспитания АПН СССР. При проведении эксперимента предусмотрен достаточно строгий медицинский контроль за состоянием здоровья детей. Результаты эксперимента дают основание утверждать, что применение компьютеров способствует развитию логического мышления детей, активизирует их инициативу и самостоятельность, делает их более подготовленными к дальнейшему систематизированному обучению.

С 1985 г. в общеобразовательных школах введен предмет «Основы информатики и вычислительной техники», главная задача которого — заложить первоначальные сведения об информатике, подготовить пользователя ЭВМ, знакомого с возможностями и сферами применения ВТ.

Дальнейшее повышение уровня знаний в области информатики осуществляется в профессионально-технических училищах, средних специальных и высших учебных заведениях, на факультетах повышения квалификации, где обучаемым читаются базовый и специальный курсы в области информатики и ВТ в соответствии с профилем подготовки.

В последние годы средства ВТ находят достаточно широкое применение при изучении целого ряда общеобразовательных и инженерных дисциплин. ВТ используется в вузах при проведении расчетов, подготовке текстовой и графической информации, построении чертежей, обработке результатов лабораторных опытов, моделировании различных процессов и явлений. Средства ВТ начинают использоваться при подготовке специалистов гуманитарного профиля. В средних учебных заведениях использование ВТ для этих целей только начинается.

В рамках новых информационных технологий обучения компьютеры используются в трех основных направлениях:

как индивидуальный инструмент, позволяющий выполнять рутинные вычислительные работы;

как персональное автоматизированное рабочее место, дополненное средствами для проведения исследований и проектирования в проблемных областях;

как автоматизированный обучающий комплекс или тренажер для закрепления знаний.

Достаточно широко в вузах используются различного рода обучающие курсы и контролирующие программы. В профессионально-технических училищах и средних специальных учебных заведениях в большей степени находят применение программы-тренажеры.

Информатизация образования невозможна без перехода в учебно-воспитательном процессе на НИТ.

НИТ в образовании есть комплекс учебных и учебно-методических материалов, технических и инструментальных СВТ учебного назначения, а также система научных знаний о роли и месте СВТ в учебном процессе, о формах и методах их применения для совершенствования труда преподавателей и учащихся.

В настоящее время делаются первые шаги, разрабатывается методология использования НИТ, создаются и совершенствуются средства НИТ, складывается практика их использования в учебных заведениях. Отчасти это объясняется тем, что необходимые для внедрения НИТ технические и программные средства стали доступны учебным заведениям лишь в последние годы. Ряд НИТ находятся в стадии становления и доступны лишь для проведения экспериментальной работы (электронная почта, интерактивное видео и т. п.).

Вместе с тем следует отметить, что в вопросах информатизации процесса обучения наметилась достаточно устойчивая тенденция перехода от использования в учебном процессе обучающих программ по отдельным разделам курса к созданию и практическому

внедрению комплексов учебно-методического и программного обеспечения, «закрывающего» полностью весь курс. Такому подходу во многом способствует создание ряда компьютеризированных учебников и учебных пособий.

НИТ в обучении представляют мощное средство повышения производительности умственного труда, позволяющее найти кардинальные решения насущных педагогических проблем и обеспечить оптимальное управление учебным процессом.

НИТ являются тем инструментом, который позволит педагогам качественно изменить методы и организационные формы своей работы, полнее сохранять и развивать индивидуальные способности обучаемых, усилить междисциплинарные связи в обучении, осуществлять постоянное динамичное обновление организации учебного процесса.

Главная отличительная черта НИТ состоит в том, что они предоставляют практически неограниченные возможности для самостоятельной и совместной творческой деятельности преподавателей и обучаемых. Как показывают первые опыты, НИТ практически бесполезны при традиционном информационно-объяснительном подходе к обучению. В условиях НИТ меняется позиция педагога в учебном процессе. Из авторитарного носителя истины он превращается в соучастника продуктивной деятельности своих воспитанников. НИТ предполагают качественный сдвиг в отношениях между педагогами и обучаемыми.

ЭВМ — это не только мощный инструмент, но и в определенной степени третий партнер в учебном процессе, который предоставляет другим его участникам богатейшие возможности по переработке информации.

В таких условиях методическая и общекультурная составляющие в подготовке преподавателя являются решающими.

Начиная с 1985 г. ведется подготовка студентов педагогических институтов и университетов по специальности «Информатика и вычислительная техника». Подготовка осуществляется как дополнение к специальностям «Математика», «Физика», «Общетехнические дисциплины и труд».

Однако к настоящему времени задача подготовки преподавателей в области информационных технологий обучения решена далеко не полностью.

Учитывая, что любые информационные технологии хороши ровно настолько, насколько подготовлены использующие их преподаватели, Государственный комитет СССР по народному образованию придает большое значение совершенствованию подготовки преподавателей в области НИТ.

В 1989 г. под непосредственным руководством Главного методического управления высшего образования в НИИ проблем высшей школы и ИК АН УССР разработан сборник программ по дисциплине «Основы компьютерной технологии обучения» для факультетов повышения квалификации и центров компьютерной технологии обучения. Сборник имеет модульную структуру и определяет содержание подготовки для разной категории слушателей, исходя из начального уровня компьютерной грамотности и конечной цели подготовки.

Большое внимание в процессе создания средств НИТ уделяется разработке инструментальных программных средств, позволяющих по-новому строить учебный процесс, формировать у обучаемого развитое теоретическое мышление. К настоящему времени в вузах создан достаточно широкий набор типовых программных средств, ориентированных на конструирование и технологическую поддержку автоматизированных учебных курсов, баз знаний и данных, психолого-педагогические исследования, информационно-справочное обслуживание и т. п.

Разработанные программные средства функционируют в среде типовых операционных систем, рассчитаны на пользователя с минимальной подготовкой, обеспечивают интерактивный режим взаимодействия с пользователями и имеют режим обучения.

К числу таких пакетов следует отнести системы СПОК и СПОК-ВУЗ (ИК АН УССР, НИИВШ), АТОС

(БГУ), ЭКСТЕРН (МГУ), ОСКАР (КазаньВТИ), АОС ВУЗ (НИИВШ), Терминал (ЧПИ), АСТАА (МИЭТ), МИДОС (МЭИ), РАКУРС (МИИГА), Эскиз (НИИВШ).

Наибольшее распространение в стране получили автоматизированные обучающие системы (АОС) семейства АОС ВУЗ.

Отечественные авторские языки и системы семейства АОС ВУЗ находятся, по нашему мнению, на уровне современных зарубежных систем для соответствующего класса технических средств. Однако наиболее перспективным направлением на сегодня является развитие экспертных систем.

В области графических и видеосредств следует отметить достаточно серьезное отставание от требований сегодняшнего дня.

Основным средством подготовки автоматизированных учебных курсов в настоящее время являются проблемно-ориентированные языки — языки описания курсов и авторские системы, включающие новые элементы систем программирования: текстовые и графические редакторы, базы данных, языки моделирования и др., а также специальные средства описания стратегий обучения и учебного материала. Намечается тенденция привлечения достижений, полученных в рамках работ по искусственному интеллекту и экспертным системам.

В вузах страны создано более 500 обучающих программ по различным дисциплинам: программированию, математике, прикладным наукам и технике, химии, моделированию, электронно-вычислительной технике, экономике, финансированию и бухгалтерскому учету, физике, филологии и иностранным языкам, ботанике и зоологии и т. д. Однако не все обучающие программы отвечают требуемому уровню.

Основным направлением разработки программных средств учебного назначения в средней общеобразовательной школе до недавнего времени было обеспечение занятий по курсу ОИВТ.

В настоящее время ведутся работы по созданию и внедрению в учебный процесс обучающих программ по школьным курсам математики, физики, геометрии, химии и т. д.

На тематической выставке «Компьютеризация образования (средства, методы, передовой опыт)», закончившейся в январе 1989 г., для демонстрации было представлено более 200 пакетов программных средств. Она показала возросший уровень методического и технического мастерства разработчиков. Вместе с тем на выставке выявилось несоответствие технических характеристик учебных ЭВМ требованиям обеспечения высокой дидактической эффективности обучающих программ и программных комплексов.

Основную трудность в настоящее время представляют нехватка и низкое качество оборудования, а также несовершенство правовой и экономической основы внедрения компьютерной технологии в учебных заведениях.

В качестве перспективных работ, проводимых в настоящее время в учебных заведениях, следует отметить исследования по аналитическому решению задач, автоматическому доказательству теорем, машинному пониманию естественного языка, семантическому представлению знаний, организации вопросно-ответных систем и созданию систем оптимального управления учебным процессом.

Для решения практических задач по информатизации образования выполнен и намечен к выполнению ряд организационно-практических мероприятий.

В течение 1988 г. в системе образования был пересмотрен перечень специальностей, по которым осуществляется подготовка специалистов в средних специальных и высших учебных заведениях, увеличен выпуск специалистов по вычислительной технике, программному обеспечению, автоматизированным системам. В системе профессионально-технических училищ начата подготовка квалифицированных рабочих по эксплуатации и ремонту средств обеспечения НИТ.

Создан Общесоюзный центр новых информационных технологий, основными задачами которого являются:

координация и проведение НИР;

прогноз развития программного и технического обеспечения;

организация отраслевого фонда алгоритмов и программ учебного назначения;

повышение квалификации специалистов и преподавателей в области НИТ.

Намечено создание центров НИТ по уровням образования, а также сети региональных центров НИТ на базе ведущих вузов страны.

При Государственном комитете СССР по народному образованию в 1989 г. организован Общесоюзный научно-методический совет для решения методологических и организационных проблем информатизации образования.

Проведен ряд мероприятий по систематизации научных исследований в области информатизации образования. Сконцентрированы усилия вузов по разработке учебного, методического и программного обеспечения в области НИТ.

Организована совместная работа с вузами социалистических стран в рамках комплексной программы научно-технического прогресса стран — членов СЭВ. В ряде случаев такая работа проводится на договорной основе.

Одной из важнейших компонент НИТ в обучении являются технические средства: компьютеры, видеосистемы, компактные диски и др. В настоящее время в СССР на государственном уровне разработаны планы, в которых намечено организовать массовый выпуск персональных компьютеров, расширить масштабы применения современных ЭВМ всех классов, повысить эффективность вычислительных центров коллективного пользования, сетей обработки и передачи информации.

В ближайшей перспективе предполагается обеспечить потребности учебных заведений в ВТ для реализации НИТ в обучении. В текущей пятилетке в высших учебных заведениях будет создано примерно 130 тыс. рабочих мест, оборудованных персональными компьютерами и терминальными устройствами, будут расширены работы по организации межвузовских сетей коллективного пользования, банков информации, улучшено информационное обслуживание процесса обучения и научных исследований.

Это создаст предпосылки для решения предусмотренной Основными направлениями перестройки высшего образования в СССР задачи — обеспечить в период обучения студентов необходимым количеством машинного времени в зависимости от специальности обучения, обеспечить ежегодное расширение объема технических и поисковых исследований в учебных заведениях, имея в виду к концу текущей пятилетки увеличить их объем не менее чем в два раза.

Получит дальнейшее развитие оснащение средних учебных заведений комплексами учебной вычислительной техники (КУВТ), специально разрабатываемых для этих целей промышленностью.

Принимаются меры по повышению качества и надежности работы учебных персональных ЭВМ.

Следует отметить, что разнообразие имеющихся и проектируемых в СССР средств ВТ затрудняет возможность построения типовых автоматизированных рабочих мест, ориентированных на использование в учебном процессе. Определение перспективы для создания НИТ связано у нас с использованием интеллектуальных терминалов (включая многоэкранные) и видеокомпьютерных систем, широкий выпуск которых планируется промышленностью в ближайшее время. Многогранность применения НИТ в обучении, разнообразие организационных форм определяют необходимость иметь широкий набор функциональных модулей, которые позволят комплектовать и специализировать рабочие места в соответствии с требованиями учебного процесса.

Эта задача решается в основном на базе серийной отечественной ВТ, главным образом за счет использования персональных ЭВМ, стандартных конструктивов для расширения функциональных возможностей рабочего места, а также штатного системного программного обеспечения. В то же время производятся закупки отдельных партий персональных компьютеров за рубежом.

Для реализации возможности использования типового программного обеспечения и стандартного доступа к базам данных и знаний создаются сети ЭВМ,

объединяющие компьютеры разных классов.

Стратегия оснащения учебных заведений СССР средствами ВТ строится в зависимости от видов обучения и профиля подготовки, с учетом меняющейся потребности народного хозяйства в специалистах.

Анализ результатов информатизации образования в нашей стране, а также внимательное изучение опыта информатизации образования в странах ЮНЕСКО позволяют определить дальнейшие направления работ в этой области. К числу таких направлений относятся следующие:

развитие научно-исследовательских и опытно-конструкторских работ в области информатизации образования, более широкое использование вузовских ученых в создании и совершенствовании НИТ;

совершенствование организационно-экономического механизма информатизации образования;

обеспечение современной технической базы НИТ, широкое использование средств спутниковой связи, развитие систем электронной почты, телетекса, видеотекса и др.;

развитие и использование НИТ в создании и обновлении учебно-методической базы (создание электронных учебников, баз знаний и т. д.);

создание и интенсивное использование центров и клубов информатизации, широкое использование периодической печати для освещения проблем компьютеризации;

развитие методов и средств информатизации в организации досуга школьников и студентов (игровые программы, деловые игры с использованием современных психолого-педагогических основ обучения);

повышение престижа разработчиков НИТ, присуждение ученых степеней за исследования в этой области;

активное включение в мировой научно-технический процесс разделения интеллектуального труда на основе взаимного обмена знаниями и информацией.

Перечисленные направления раскрывают тот большой и разносторонний объем деятельности по созданию НИТ, который во многих случаях может быть выполнен только коллективом или коллективами разработчиков, т. е. вопрос идет об интеграции усилий всех заинтересованных сторон, которая позволит создать современные средства обеспечения учебного процесса, отвечающие требованиям подготовки высококвалифицированных, творчески мыслящих специалистов.

Верить — не верить?

Все чаще для доказательства математических утверждений используются ЭВМ. Именно с ее помощью было показано, например, что любую, самую причудливую, географическую карту можно раскрасить четырьмя красками так, что любые две соседние страны будут иметь разные цвета (теорема о четырех красках). Однако не только решения, но и проблемы порождает применение компьютеров.

Затратив 3000 ч машинного времени супер-ЭВМ «Крэй-1», группа математиков объявила, что не существует конечных проективных плоскостей 10-го порядка. Если бы такая плоскость была найдена компьютером, можно было бы проверить, истинна ли находка. Но «Крэй-1» решения не нашел, и его «работодатели» считают, что этим доказано отсутствие решения.

Правы ли они? Нет ли ошибки в программе, полон ли перебор вариантов? Проверка обойдется очень дорого и может ничего не дать, если пресловутая плоскость снова не будет найдена.

Остается доверять авторитету ученых.

Это интересно

План первых трех лет пятилетки по поставкам вычислительной техники в школы выполнен на 57,5 %.

С 25 по 27 апреля 1989 г. в г. Воронеже состоялся учредительный пленум Общесоюзного научно-методического совета информатизации образования.

Решение пленума

Общесоюзного научно-методического совета информатизации образования

1. Учредить Общесоюзный научно-методический совет информатизации образования, решающий в своей деятельности задачи перестройки народного образования на основе разработки и внедрения новых информационных технологий в деятельность учебных заведений.

2. Одобрить в целом Положение об Общесоюзном научно-методическом совете информатизации образования и поручить Главному ученому секретарю Совета с учетом высказанных предложений и замечаний доработать его и в месячный срок представить на утверждение в Гособразование СССР.

3. Предложенный состав Совета и его руководящий состав представить для утверждения в Гособразование СССР.

4. Направить для утверждения в Гособразование СССР план работы Общесоюзного научно-методического совета информатизации образования на 1989—1990 гг.

5. Считать целесообразным создание при Гособразовании СССР экспертного совета по новым информационным технологиям в образовании с возможностью финансирования экспертных работ.

6. Создать рабочие группы:

по подготовке проекта Концепции информатизации образования;

по осуществлению технической политики в области средств новых информационных технологий в образовании, проведению экспертизы средств вычислительной техники, в том числе эргономической и психолого-педагогической.

7. Обратите внимание Гособразования СССР на недопустимость поставки в учебные заведения морально устаревших средств вычислительной техники катастрофически низкого качества.

8. Просить Гособразование СССР:

впредь именовать Совет — Общесоюзным научно-методическим советом информатизации образования;

подготовить предложения в Совет Министров СССР о возложении функций основного заказчика средств новых информационных технологий в образовании на Гособразование СССР;

выйти в Совет Министров СССР с предложением о передаче всех средств, выделяемых на информатизацию образования, Гособразованию СССР с целью исключения дублирования и распыления средств финансирования всех работ в стране по информатизации образования;

создать резерв материальных ресурсов для обеспечения по рекомендации Совета наиболее перспективных разработок в сфере новых информационных технологий;

подготовить предложения в Совет Министров СССР о создании в структуре Гособразования СССР постоянно действующего подразделения — координирующего органа по управлению системой информатизации образования.

Положение

об Общесоюзном научно-методическом совете информатизации образования Гособразования СССР

1. Общие положения

Общесоюзный научно-методический совет информатизации образования Государственного комитета СССР по народному образованию, в дальнейшем именуемый — Совет, является общественным органом, определяющим направления в разработке и реализации новых информационных технологий (НИТ) в сфере образования.

Состав Совета и срок его полномочий определяются решением коллегии (приказом) Гособразования СССР.

Свою деятельность Совет направляет на разработку, развитие и реализацию

концепции информатизации образования как составной части концепции информатизации общества.

В своей деятельности Совет руководствуется решениями партии и правительства по вопросам информатизации общества, в том числе народного образования, приказами и решениями коллегии Гособразования СССР, настоящим Положением.

2. Цели деятельности Совета

Деятельность Совета направлена на перестройку народного образования на основе его информатизации в соответствии с государственной политикой информатизации общества.

Совет осуществляет: разработку, последующее развитие концепции информатизации образования и способствует ее реализации; участвует в формировании структуры системы информатизации образования с учетом правовых и экономических взаимоотношений.

3. Основные функции Совета

Совет:

разрабатывает рекомендации Гособразованию СССР по поэтапному внедрению НИТ с учетом тенденции развития по основным направлениям создания программно-методического обеспечения учебного процесса;

изучает состояние преподавания, организацию и результаты научно-исследовательской и научно-методической работы в учебных заведениях с точки зрения использования средств НИТ, разрабатывает рекомендации по совершенствованию подготовки, переподготовки и повышению квалификации руководящих работников и специалистов народного хозяйства;

анализирует, обобщает и распространяет передовой отечественный и зарубежный опыт использования НИТ в народном образовании;

оценивает эффективность использования НИТ;

осуществляет экспертизу планов и результатов внедрения НИР и ОКР;

оказывает научно-методическую помощь учреждениям и учебным заведениям, в том числе на хозрасчетной основе;

широко освещает результаты своей деятельности в средствах массовой информации.

4. Структура и состав Совета

Высшим органом Совета является пленум. Все члены Совета пользуются на пленуме равными правами. В перерывах между пленумами работой Совета управляет избираемый пленумом руководящий состав, возглавляемый председателем. Председатель и руководящий состав Совета, так же как и полный состав Совета, утверждаются решением коллегии Гособразования СССР.

Совет состоит из следующих секций: **координации; дошкольного воспитания; школ; профессионально-технического образования и внешкольных учреждений; средних специальных учебных заведений; высших учебных заведений; подготовки и повышения квалификации педагогических кадров.**

Совет создается на общественных началах из преподавателей и специалистов учебных заведений, учреждений системы повышения квалификации, научных учреждений, органов управления народным образованием и, при необходимости, специалистов других ведомств.

Состав Совета формируется по предложениям научно-педагогической общественности с учетом рекомендаций союзного и республиканских органов управления народным образованием и утверждается приказом Гособразования СССР. Количественный состав Совета определяется в каждом конкретном случае, исходя из особенностей деятельности секций.

5. Организационные формы работы

Работа Совета осуществляется по перспективному и годовому планам, согласованным с Гособразованием СССР.

Перспективные планы включают разработку вопросов, исходя из задач, определенных настоящим Положением. Годовые планы составляются на основе перспективных с учетом текущих вопросов.

Планы работы секций Совета в основном являются детализированными, с учетом своей специфики, составными частями общего плана.

Пленарные заседания Совета проводятся не реже одного раза в год. Заседания секций проводятся в соответствии с планом работ не реже двух раз в год.

По предложению Совета для осуществления целевых программ или решения отдельных проблем (задач) могут создаваться временные научно-исследовательские коллективы и рабочие группы, в том числе на основе целевого финансирования Гособразованием СССР, а также на хозрасчетной основе.

Командирование членов Совета и рабочих групп осуществляется администрацией организации, в штате которой работает данный специалист.

Решения Совета или секции принимаются большинством голосов открытым голосованием и оформляются протоколом, который подписывается и хранится у ученого секретаря.

Вопросы, затрагивающие интересы несколько секций или являющиеся общими для них, рассматриваются секцией координации.

Совет должен систематически информировать научно-педагогическую общественность о результатах своей работы.

6. Права Совета

Совет имеет право:

проводить экспертизу состояния внедрения НИТ в образовании; заслушивать сообщения представителей территориальных органов управления народным образованием, учебных заведений, учебно-методических объединений, а также, по согласованию с Гособразованием СССР, представителей отраслей о ходе выполнения этих работ;

заслушивать руководителей научно-исследовательских организаций, занимающихся применением НИТ в образовании, по вопросам планирования, ходу выполнения, результатам внедрения НИР и ОКР;

разрабатывать рекомендации по целевому финансированию разработок в области НИТ и заслушивать их отчеты о ходе выполнения этих работ и полученных результатах;

представлять рекомендации по составу авторских коллективов для создания учебных программ, учебников, учебных пособий и других учебно-методических материалов и давать заключения о целесообразности публикаций; обсуждать проспекты, перспективные планы изданий и изданную литературу;

организовывать работу по созданию учебно-методического обеспечения на правах учебно-методического объединения по НИТ;

вносить предложения о создании временных научно-технических коллективов, рабочих групп для решения отдельных задач; о командировании членов Совета и привлекаемых специалистов для изучения отечественного и зарубежного опыта, по материально-техническому обеспечению средствами НИТ учебных заведений;

участвовать в организации и проведении международных, всесоюзных, республиканских и региональных совещаний, симпозиумов, семинаров, конференций по вопросам организации и содержания учебно-воспитательного процесса и научно-исследовательской работы и обмена опытом в области НИТ; в разработке и экспертизе комплексных целевых программ;

изучать опыт работы учебных заведений, состояние подготовки, переподготовки и повышения квалификации педагогических и научно-педагогических кадров по специальностям в соответствии со спецификой деятельности секций Совета;

вносить в Гособразование СССР предложения по распределению целевого резерва средств НИТ.

7. Права и обязанности членов Совета

Члены Совета имеют право:

выносить на обсуждение Совета (секции) вопросы, связанные с задачами и содержанием работы Совета;

присутствовать на всех заседаниях Совета (секции) и участвовать в принятии решений;

быть командированными в установленном порядке по вопросам, находящим-

ся в компентенции Совета;

быть представленными в установленном порядке к поощрению по рекомендации Совета или секции.

Члены Совета обязаны:

активно участвовать в работе секции;

выполнять поручения и задания руководства Совета или секции;

способствовать максимальному распространению решений и рекомендаций Совета;

проводить политику Совета на местах.

Прием новых членов и изменение состава Совета проводятся на пленарных заседаниях по предложению руководящего состава Совета.

Время, затрачиваемое членом Совета (секции) на работу в активе Совета, учитывается при определении различных видов поручений, включая учебные, фиксируется в индивидуальном плане по месту его основной работы.

В. Г. Житомирский

Сегодня мы расскажем о вкладе в развитие школьной информатики Владимира Габриэлевича Житомирского (1934—1988) — видного деятеля математического образования, доцента, кандидата физико-математических наук, заведующего кафедрой информатики и вычислительной техники и директора вычислительного центра Свердловского педагогического института.

В. Г. Житомирский в 1957 г. окончил Уральский государственный университет, затем несколько лет работал учителем средней школы в Свердловске, преподавателем Уральского электромеханического института инженеров железнодорожного транспорта. Первоначально областью его научных интересов была алгебра; в 1963 г. он успешно защитил кандидатскую диссертацию, посвященную группам автоморфизмов групп.

В дальнейшем вся деятельность В. Г. Житомирского была связана со Свердловским пединститутом. Владимир Габриэлевич был замечательным педагогом — превосходным лектором и тонким методистом, подлинным мастером своего дела, артистичной натурой. Он быстро завоевал авторитет в коллективе института. После разделения физмата на два факультета он стал первым деканом математического факультета. На этом посту особенно проявились его качества талантливого организатора учебного процесса, доброжелательного руководителя педагогического коллектива, мудрого наставника студентов. За годы работы в педагогическом институте В. Г. Житомирский вырос в крупного деятеля просвещения.

Особенно велик вклад В. Г. Житомирского в дело компьютеризации народного образования. Обладая незаурядным общественным темпераментом и чутко ощущая дух времени, он серьезно заинтересовался этими вопросами еще в середине 60-х гг. Он был одним из инициаторов создания в пединституте вычислительного центра, работой кото-

рого руководил с 1966 г. В 1976 г. Владимир Габриэлевич возглавил созданную благодаря его усилиям кафедру вычислительной математики, преобразованную впоследствии в кафедру информатики и вычислительной техники. В 1985 г. он стал научным руководителем новой проблемной лаборатории по использованию электронно-вычислительной и микропроцессорной техники в учебно-воспитательном процессе средних общеобразовательных школ и педагогических институтов.

Деятельность В. Г. Житомирского в области компьютеризации образования отличалась исключительной целеустремленностью и плодотворностью. Он был инициатором многих начинаний, автором или редактором большого числа статей, методических и учебных пособий, сборников научных трудов. Всего им опубликовано более 130 работ, список которых венчают написанные совместно с В. М. Заварыкиным и М. П. Лапчиком учебные пособия для педагогических институтов (с грифом Минпроса СССР) «Вычислительная математика» (Свердловск, 1985), «Лабораторный практикум по вычислительной математике» (Свердловск, 1986), «Алгоритмы и вычисления в курсе математики» (Свердловск, 1986), «Техника вычислений и алгоритмизации» (М.: Просвещение, 1987), «Использование вычислительной техники в учебном процессе» (Свердловск, 1987), «Основы информатики и вычислительной техники» (М.: Просвещение, 1988), а также написанное совместно с А. Г. Гейном, Е. В. Линецким, М. В. Сапиром и В. Ф. Шолоховичем экспериментальное учебное пособие для средней школы «Основы информатики и вычислительной техники» (Свердловск: Изд-во Урал. ун-та, 1989). В. Г. Житомирский принимал участие в разработке утвержденных Минпросом СССР программ для физико-математических специальностей в пединститутах по учебным курсам «Техника вычислений и алгоритмизация», «Основы информатики и вычислительной техники», «Численные методы», «Методика преподавания информатики», «Использование вычислительной техники в учебном процессе». При его непосредственном участии в 1982 г. был создан экспериментальный учебный план по специальности «Математика и управление учебным процессом на базе ЭВМ», с 1983 г. впервые в стране подготовка студентов по этой специальности начата в Свердловском пединституте. С созданием Свердловского областного координационного научно-методического совета по компьютеризации В. Г. Житомирский стал заместителем председателя этого совета.

В значительной степени благодаря творческой и организаторской деятельности В. Г. Житомирского Свердловск стал одним из признанных центров по использованию вычислительной техники в народном образовании. Признанием роли Свердловского пединститута в этой области явилось, в частности, проведение на его базе трех всесоюзных совещаний-семинаров по проблеме «Применение ЭВМ в обеспечении учебного процесса и управлении образованием» (1974, 1979 и 1984 гг.). Свердловский пединститут является головной организацией по комплексной научно-исследовательской программе «Содержание, организация и методика изучения информатики и вычислительной техники в общеобразовательной школе, педагогических учебных заведениях и аппарате управления», в реализации которой участвуют 12 пединститутов страны. Научным руководителем этих исследований был В. Г. Житомирский. Под его руководством выполнены и успешно защищены две кандидатские диссертации, несколько диссертаций близки к завершению. Много заботы вложил Владимир Габриэлевич в создание и организацию работы городского школьного компьютерного центра и нескольких десятков межшкольных кабинетов информатики, часто выступал с лекциями и докладами перед учителями и другими работниками народного образования. Он был последовательным и поистине неутомимым пропагандистом комплексного использования вычислительной техники в системе образования[1].

[1] О созданном в Свердловске целостном программно-методическом комплексе, обеспечивающем учителей и учащихся всем, что необходимо для эффективного усвоения курса информатики

Трудно охарактеризовать все виды многогранной деятельности В. Г. Житомирского. В течение многих лет он руководил в институте методологическим семинаром математического факультета. Он был членом комиссии по информатике Ученого методического совета Министерства просвещения СССР, членом межведомственной комиссии по разработке проекта концепции использования средств вычислительной техники в сфере образования, членом объединенного совета по математике и механике Уральского отделения АН СССР. Ни одну из своих многочисленных обязанностей Владимир Габриэлевич не исполнял формально, в любое дело он вкладывал душу, щедро отдавая знания, опыт и, казалось, неистощимую энергию.

Особо следует сказать о многолетней творческой работе Владимира Габриэлевича по созданию популярных математических книг для детей младшего возраста. Написанные им совместно с Л. Н. Шевриным книги «Геометрия для малышей» (1-е изд.— 1969 г.) и «Математическая азбука» (1-е изд.— 1980 г.) неоднократно выходили на русском языке, переведены на многие иностранные языки и языки союзных республик и получили широкую известность в нашей стране и за рубежом. На основе первой книги были написаны сценарии трех мультфильмов, снятых на Свердловской киностудии: «Путешествие по стране Геометрии» (1972), «Снова в стране Геометрии» (1976) и «Кому нужна Геометрия» (1981).

Заслуги В. Г. Житомирского отмечены медалями «За доблестный труд. В ознаменование 100-летия со дня рождения В. И. Ленина» и «За трудовую доблесть», значком «Отличник народного просвещения», тремя медалями ВДНХ, многочисленными грамотами и благодарностями.

Человеком редкого таланта, широких интересов и активной жизненной позиции, настойчивым и принципиальным в делах, вдумчивым и чутким руководителем и воспитателем, неизменно корректным по отношению к коллегам и демократичным со студентами, живым и остроумным собеседником — таким останется Владимир Габриэлевич Житомирский в памяти тех, кому довелось с ним общаться.

Н. КРАСОВСКИЙ,
Б. СУТЫРИН, Л. ШЕВРИН

(в машинном варианте), и в особенности о методическом компоненте этого комплекса, рассказано в написанной В. Г. Житомирским с соавторами обстоятельной статье «Программно-методический комплекс по курсу школьной информатики», опубликованной в журнале «Информатика и образование» (№ 3 и 4 за 1988 г.).

Компьютер в музыкальной школе

В Йенской музыкальной школе (ГДР) ЭВМ используется в качестве музыкального тренажера на уроках духовой музыки. Компьютер по несложной программе воспроизводит на мониторе ноты выбранного музыкального упражнения и, если это необходимо, соответствующий им звуковой эквивалент. Юные музыканты, читая ноты с экрана, играют предлагаемое компьютером произведение, сверяя свою трактовку музыкального текста с «сухой», но точной компьютерной.

Такая методика позволяет завершить обучение нотной грамоте всего за 4 занятия. Традиционная форма познания музыкальных азов требует 6 месяцев при 2 занятиях в неделю. Многократно возрастающая интенсивность обучения вместе с тем не утомляет ребят, для которых проверка гамм превращается в увлекательную игру с компьютером.

ЭВМ может

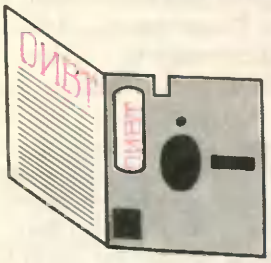
Поможет термопринтер

Примерно восемь точек на миллиметр — такова точность изображения, которое можно получить при помощи термопечатающего устройства TDU-850 фирмы «Рейтеон». Яркость цвета каждой точки может иметь 64 градации, а при использовании специального режима «Суперпиксел» на бумаге можно отобразить 256 уровней серого цвета.

Качество исходной продукции позволяет применять термопринтер для распечатки компьютерных программ или текстов почти типографским шрифтом, использовать его в комплексах обработки изображений, системах автоматизации проектирования печатных плат, телефаксах.

Да, конечно, лазерный принтер обеспечивает еще более высокое качество, чем термопечатающее устройство TDU-850, кстати требующее специальной термобумаги, но ведь это право пользователя — выбирать между небольшой потерей качества и крупной тратой наличных.

Методика обучения



А. ГЕЙН, Е. ЛИНЕЦКИЙ, М. САПИР, В. ШОЛОХОВИЧ

Информатика: алгоритмические конструкции

I. Дискуссии о месте алгоритмизации в обучении информатике начались с массовым обучением этому предмету и не утихают до сих пор. К настоящему времени сформировались две крайние точки зрения (и много промежуточных). Одна из них, «программистская», состоит в том, что информатика — это программирование и потому алгоритмизация занимает в ней центральное место. Другая — «пользовательская» точка зрения: информатика сводится к обучению пользованию готовыми программными средствами (информационными системами, электронными таблицами и т. д.) и алгоритмизация не нужна. Нам кажется, что вопрос о роли алгоритмизации должен решаться исходя из основных целей обучения в школе вообще. Одна из таких целей — развитие мышления учащихся, которое предполагает:

1) создание у школьников целостной картины мира;

2) формирование навыков жизни в информатизированном обществе;

3) развитие способностей к общению, коллективной деятельности;

4) активизацию процесса самопознания.

Каждый школьный предмет призван прежде всего внести свой вклад в достижение указанной цели (на своем материале и своими средствами). Это относится и к отдельной теме школьного курса. Более того, требование, чтобы тема работала на достижение этой цели,— основное при решении вопроса, включать или не включать ее в школьный курс. Раздел «алгоритмизация» курса ОИВТ, безусловно, удовлетворяет этому требованию.

Картина мира будет неполной без ясного представления о том, что все процессы, протекающие в мире, так или иначе основаны на выполнении алгоритмов. Понимание того, что в основе любого алгоритма лежат три конструкции — *последовательное выполнение, развилки и циклы*, не менее важно, чем знание атомно-молекулярной теории строения вещества. Отметим, что всеобщность понятия «алгоритм» вовсе не означает жесткую детерминированность процессов. Показ такой недетерминированности необходим и вполне доступен для учащихся (примером может служить изложение метода Монте-Карло в [1]). Обсуждение вероятностных моделей и соответствующих алгоритмов чрезвычайно важно для формирования адекватной картины мира.

Жизнь в современном обществе невозможна, если не умеешь планировать свои действия и эффективно учиться. При правильной постановке курс алгоритмизации способен внести большой вклад в формирование этих умений. Особая роль принадлежит методу пошаговой детализации, ибо без него невозможно спланировать сколь-нибудь сложную последовательность действий. Поэтому, на наш взгляд, метод пошаговой детализации должен пронизывать весь курс ОИВТ, а не быть просто одной из многих изучаемых тем. И хотя тема «Пошаговая детализация» далеко не первая в курсе ОИВТ, целесообразно организовывать разбор задач так, чтобы школьникам приходилось фактически использовать этот метод.

Метод пошаговой детализации может и должен использоваться и в других школь-

---

* Материал статьи охватывает семь параграфов учебника [1] — с седьмого по тринадцатый.

Продолжение. Начало см.: Информатика и образование. 1989. № 2, 3, 4.

ных курсах. Так, В. Г. Болтянский [2] отмечает, что применение метода пошаговой детализации, на котором базируется поиск пути решения, в наиболее полном виде соответствует идее внесения алгоритмизации в курс математики. В то же время само обучение пошаговой детализации не может основываться на решении задач из стандартных курсов математики и физики. Дело в том, что пошаговая детализация служит лишь организующим началом поиска пути решения, но далеко не сводится к нему. В этот поиск входит и актуализация необходимых фактов, законов, теорем и т. д. Поэтому в большинстве задач из обычных школьных курсов пошаговая детализация состоит не более чем из двух шагов, причем второй шаг — доведение до вычислительного алгоритма — обычно не представляет интереса с точки зрения обучения соответствующему курсу. Сами алгоритмы при этом, как правило, линейны и содержат не более шести-семи действий. Таким образом, обучение методу пошаговой детализации — самостоятельная цель алгоритмизации, не достигаемая в других школьных предметах.

В отличие от планирования «сверху вниз» обучение всегда происходит «снизу вверх»: полученные знания, умения, навыки «свертываются» во вспомогательные алгоритмы и в дальнейшем выступают в роли элементарных действий. Действительно, хорошо известно [3], что процесс формирования умственных действий обучаемого обязательно проходит целый ряд этапов. Этот процесс начинается с полностью развернутого материализованного действия и заканчивается наиболее общей, сокращенной и автоматизированной формой умственного действия. Построение обучения тем самым сводится либо к предъявлению (в так называемом втором типе обучения) учащемуся, либо построению им самим по существенным признакам (в третьем типе обучения) алгоритма действия. При этом свертывание такого алгоритма происходит как раз за счет выделения уже отработанных фрагментов, выступающих затем как одно действие, конечный продукт которого появляется сразу. Овладение учащимися алгоритмической культурой позволяет им самим осознанно строить обучение и самообучение по третьему типу, добиваясь высокой эффективности такого обучения.

Метод пошаговой детализации играет важную роль в развитии способностей к общению, коллективной деятельности. Он позволяет организовать коллективное решение школьниками сложных задач (см., например, лабораторную работу 8 из [1]). При этом отрабатываются следующие навыки:

разделение действий между разными участниками и их кооперация при решении задачи;

осознание и учет одним учащимся способа решения задачи, осуществленного другим;

взаимный контроль и оценка действий учащихся в ходе решения задачи.

Таким образом, алгоритмизация позволяет учителю вести учебную деятельность в форме коллективного диалога, т. е. проектировать среду обучения как динамически развивающуюся общность учащихся. Это дает в руки учителю мощный инструмент подготовки школьников к жизни в коллективе. Отметим, что подобные коллективистские навыки могут воспитываться и на других школьных предметах. Пока, однако, лишь на уроках труда можно встретить нечто похожее на пошаговую детализацию задачи.

Как известно, одним из средств активизации самопознания является ролевая игра. Попытка представить себя в роли другого заставляет человека и на себя взглянуть со стороны. Однако навык к исполнению роли весьма непрост и должен специальным образом формироваться. Ролевое исполнение и составление алгоритмов способствуют приобретению такого навыка. Этому помогает и составление так называемых *протоколов выполнения алгоритма*, когда учащиеся мысленно совершают действия алгоритма и комментируют их. Отметим, что таков один из возможных путей реализации «громкоречевого» этапа формирования умственных действий. Таким образом, умение составить протокол выполнения алгоритма — базовое умение, которое должно быть выработано у школьника.

II. Каковы же основные цели раздела «алгоритмизация»? На наш взгляд, школьников нужно научить:

1) основным способам организации действий в алгоритмах;

2) основным способам организации данных;

3) применению алгоритмических конструкций при составлении алгоритмов решения разнообразных классов задач.

Главное при достижении первой из этих целей — добиться понимания того, что все многообразие способов организации действий базируется на конечном числе алгоритмических конструкций. Число реализованных конструкций является конечным в любом алгоритмическом языке. Поэтому выбор той или иной системы базовых алгоритмических конструкций

OldPC.su
7002

(и даже создание нового языка) диктуется чисто методическими соображениями. Естественно, что у разных авторов эти соображения могут различаться. Например, в [4] базовыми конструкциями языка являются: последовательное выполнение, ветвление «если» (в полной и неполной формах), ветвление «выбор», ветвление «контроль» (команда «утв»), цикл «пока», цикл «повторить N раз», цикл «для каждого», вспомогательный алгоритм, алгоритм-функция (возможно, мы что-то упустили). Такое разнообразие изучаемых базовых конструкций продиктовано, по-видимому, установкой авторов на обучение программированию и стремлением собрать в одном универсальном языке программирования наиболее популярные алгоритмические конструкции.

Мы, во-первых, не считаем обучение программированию центральной целью курса ОИВТ. Во-вторых, в курсе алгоритмизации, как и в других разделах нашего учебника, мы стараемся придерживаться принципа минимальной достаточности методических средств. Поэтому «устройства управления» наших исполнителей (Чертежника, Вычислителя, Робота-манипулятора) допускают всего три базовых конструкции — ветвление (в полной и неполной формах), цикл «пока» и вспомогательный алгоритм. Цикл «для каждого» может быть реализован только для Вычислителя, поскольку лишь он умеет оперировать с числами и числовыми переменными. При этом цикл «для каждого» рассматривается как разновидность цикла «пока».

III. Укажем основные идеи, которые должны быть донесены до школьников при изучении алгоритмических конструкций.

1. Ветвления.

1а. Ветвление в алгоритмах появляется тогда, когда испонителю необходимо сделать выбор одного из нескольких наборов действий в зависимости от некоторого условия.

1б. Проверка этого условия должна являться допустимым действием исполнителя (в частности, если исполнитель не умеет проверять условия, то для него нельзя писать разветвляющиеся алгоритмы).

1в. Существует две формы ветвлений — полная (имеющая две ветви) и неполная (имеющая одну ветвь).

1г. Любой выбор можно свести к одному или нескольким ветвлениям.

1д. При записи ветвлений необходим указатель, отделяющий ветвление от остальной части алгоритма. При отсутствии такого указателя алгоритм становится двусмысленным.

1е. При решении жизненных задач ветвления возникают неизбежно. В основе многих моделей лежат различные неравенства и другие соотношения между исходными данными и результатами, которые в алгоритме воплощаются в форме ветвлений. И вообще ограниченность сферы применения любой модели приводит к появлению в соответствующем алгоритме того, что исходные данные удовлетворяют ограничениям, при которых математическая модель соответствует реальности.

2. Циклы.

2а. Появление циклов в алгоритме обусловлено необходимостью повторять определенный набор действий до тех пор, пока выполняется некоторое условие.

2б. Цикл «пока» может выполнять любой исполнитель, который «умеет» проверять условия (цикл «для каждого» доступен только для тех исполнителей, которые «умеют» работать с числами и числовыми переменными).

2в. Условие продолжения цикла проверяется только перед очередным выполнением тела цикла. Цикл прекращается лишь в том случае, когда к моменту очередного выполнения его тела условие оказывается нарушенным. В частности, если условие цикла не выполнено с самого начала, то тело цикла не исполнится ни разу. По ходу исполнения тела цикла условие может нарушиться — это не вызовет немедленного прекращения цикла.

2г. При записи цикла необходим указатель, отделяющий его тело от остальных действий алгоритма.

2д. Циклы демонстрируют эффективность применения ЭВМ: с помощью короткой циклической программы можно организовать выполнение большого количества действий.

3. Вспомогательные алгоритмы.

3а. В роли вспомогательного может выступать любой алгоритм, если его снабдить заголовком, позволяющим вызывать его из других алгоритмов.

3б. Для того чтобы алгоритм можно было вызывать из других алгоритмов, в заголовке нужно указать название, аргументы (т. е. имена тех переменных, значения которых передаются вспомогательному алгоритму из основного) и результаты (т. е. имена тех переменных, значения которых передаются из вспомогательного алгоритма основному).

3в. При вызове вспомогательного алгоритма переменные основного алгоритма

«замораживаются» (кроме результатов вспомогательного алгоритма), а после завершения вспомогательного алгоритма — «размораживаются». Таким образом, одинаково обозначенные переменные в основном и вспомогательном алгоритмах — просто «однофамильцы» (кроме результатов вспомогательного алгоритма).

3г. Результаты, ради которых основной алгоритм «обращается» к вспомогательному, далеко не всегда пригодны для дальнейшего использования исполнителем алгоритма. Это могут быть, например, сообщения, выдаваемые на экран или принтер. В подобных случаях вводят специальную переменную и ее значениями заменяют (кодируют) сообщения. Такая переменная называется сигнальной. Каждому из возможных сообщений соответствует свое значение сигнальной переменной.

3д. Создавая вспомогательный алгоритм, мы как бы добавляем еще одно допустимое действие. В результате выделения вспомогательного алгоритма подробные объяснения того, что нужно делать, можно заменить одной командой. И хотя класс достижимых целей от этого не расширяется, алгоритмы для достижения этих целей сокращаются и упрощаются (здесь просматривается глубокая аналогия с обучением человека).

3е. Вспомогательные алгоритмы могут быть составлены заранее. Составление алгоритма из вспомогательных алгоритмов подобно сборке дома из готовых блоков. Чем крупнее и универсальнее блоки, тем легче сборка.

3ж. Вспомогательные алгоритмы выступают в качестве сменных деталей алгоритмов. Для «переналадки» основного алгоритма на решение другой задачи часто достаточно заменить вспомогательный алгоритм другим вспомогательным алгоритмом, имеющим те же аргументы и результаты. Точно так же происходит переналадка сложных устройств: один блок заменяют другим, имеющим те же входы и выходы.

3з. Вспомогательные алгоритмы — этапы в пошаговой детализации алгоритмов (см. выше).

IV. Методическими установками определяется и порядок введения алгоритмических конструкций. В ряде учебников первой изучаемой конструкцией является цикл. Как правило, это конструкция «повторить N раз». Такой подход объясняется, по-видимому, тем, что появление циклической конструкции легко мотивировать сокращением записи алгоритмов, в которых некоторый набор действий повторяется, как правило, заранее известное число раз. Проверка условия в цикле «повторить N раз» практически элиминирована, и алгоритм часто продолжает восприниматься учащимися как линейный, только иначе оформленный. Платой за такое «облегченное» восприятие становятся трудности в освоении циклов как формы организации действий, качественно отличной от линейной. Во-первых, другие разновидности цикла (цикл «для каждого» и цикл «пока») воспринимаются как изолированные друг от друга, и главный признак — повторяемость действий — выступает не системообразующим, а чисто внешним: в каждой разновидности свои законы образования. Во-вторых — и это, возможно, главное — за кадром остаются опорные признаки, по которым происходит научение формированию циклов: правильное выделение условия продолжения или окончания цикла и правильное выделение тела цикла. И в случае цикла «повторить N раз», и в случае цикла «для каждого» эти два момента (выделение условия и выделение тела цикла) слабо связаны между собой, что порождает неверный стереотип у учащихся в восприятии циклов вообще. В-третьих, изучая циклы в форме «пока», учащиеся вынуждены усваивать сразу оба эти момента, что, разумеется, снижает эффективность.

Поэтому, на наш взгляд, методически более целесообразным является первоначальное изучение ветвлений, т. е. ознакомление учащихся с изменением порядка действий в зависимости от выполнения или невыполнения некоторого условия, а уже затем переход к изучению циклов.

Тема «вспомогательные алгоритмы» чрезвычайно идейно насыщена (см. п. III) и относительно более сложна, чем темы «ветвления» и «циклы». В частности, до начала изучения пошаговой детализации (неотъемлемой части темы «вспомогательные алгоритмы») школьники должны быть подготовлены к ее восприятию. Метод пошаговой детализации должен восприниматься ими как естественный метод решения задач. Этого можно достичь лишь применением этого метода при решении большого числа задач. Кроме того, выделение вспомогательных алгоритмов не является необходимой алгоритмической конструкцией: ветвлений и циклов достаточно для составления любого алгоритма. Поэтому вспомогательные алгоритмы целесообразно изучать после ветвлений и циклов.

С другой стороны, вспомогательные алгоритмы — один из способов организации действий и нецелесообразно отрывать его от ветвлений и циклов (как это сделано, к примеру, в [5]).

Высказанные соображения показывают,

что наиболее естествен следующий порядок изложения тем в разделе «алгоритмизация»:

линейные алгоритмы;

ветвления;

цикл «пока»;

вспомогательные алгоритмы;

табличный способ организации данных, цикл «для каждого».

Обратимся теперь к более детальному разбору методики изучения этих тем. Методика изложения теоретического материала в основном приведена в [6]. Все многообразие конкретных методических соображений, которыми руководствовались авторы при отборе примеров, определении порядка изложения теоретического материала в каждой из тем, выбора системы наглядных образов, упрощающих восприятие и т. п., слишком велико и физически не может быть вмещено в рамки журнальной статьи. Остановимся более подробно на методике решения задач. Напомним (см. [6]), что задачи из [1] можно условно разбить на четыре типа:

найти ошибку в алгоритме, программе, модели и т. д.;

определить, каков результат выполнения алгоритма (программы);

составить алгоритм (программу);

построить математическую модель, составить алгоритм, написать программу.

V. Решая задачи на нахождение ошибок в разветвляющихся алгоритмах и задачи типа «что произойдет в результате выполнения алгоритма», учащиеся должны научиться определять, какие условия проверяются, какие действия совершаются при выполнении условий, а какие — при невыполнении; различать ветвления в полной и неполной формах; осознать необходимость указателя «Конец ветвления».

При решении задач этих типов целесообразно составлять протоколы выполнения алгоритмов (см. п. I). Процедуры построения протоколов для разветвляющихся и линейных алгоритмов различны. Встретив ветвление, необходимо рассмотреть оба возможных случая — и случай, когда условие выполняется, и случай, когда условие не выполняется. Таким образом, протокол выполнения разветвляющегося алгоритма сам «разветвляется». Распространенная ошибка — «отслеживание» лишь одного из двух случаев. Протокол выполнения алгоритмов типа Чертежника можно представлять в виде последовательности рисунков. Такая наглядность — одно из достоинств Чертежника.

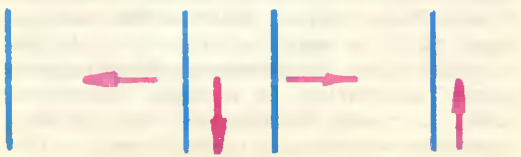
Для учащихся психологическую трудность представляет необходимость выполнять действия не в том порядке, в каком они записаны. Чтобы ее преодолеть, нужно выработать у школьников навык анализа построения алгоритма, его логической структуры (отвлекаясь от содержания действий, составляющих алгоритм). Этот анализ предусматривает ответы на вопросы типа «Есть ли в алгоритме ветвление?», «Какая форма ветвления применена?», «Из каких действий состоит первая, а из каких — вторая ветвь ветвления?» и т. д.

При решении задач на составление алгоритмов следует обсуждать с учащимися необходимость использования ветвлений. Целесообразно задавать учащимся вопросы типа «Почему алгоритм решения задачи не может быть линейным?», «Какое условие надо проверять при выполнении алгоритма?», «Какие действия надо совершать при выполнении условия, а какие — при его невыполнении?», «Где поставить указатель конца ветвления?» и т. п. Важно обращать внимание на аргументацию необходимости использования в алгоритмах тех или иных форм ветвления и на определение количества ветвлений. Это определяется числом возможных случаев, возникающих при решении задачи. Например, если может возникнуть только один из двух случаев, то можно воспользоваться одним ветвлением. Если же возможны три случая и более, то одним ветвлением не обойтись.

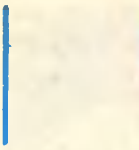
Одна из основных трудностей, с которыми сталкиваются школьники при составлении разветвляющихся алгоритмов,— алгоритм должен давать верный результат при всех допустимых исходных данных (состояниях). Преодоление этой трудности отрабатывается с помощью исполнителя Чертежник, который допускает лишь несколько легко обозримых исходных состояний: четыре состояния в углу листа, четыре состояния у края, но не в углу и одно состояние — не у края листа (см. рис. 1—3).

При составлении и анализе коротких алгоритмов целесообразно использовать блок-


1


2

3



схемы. Очевиден эффект их наглядности. Их удобно использовать и для анализа логической структуры алгоритма (достаточно «стереть» содержимое блоков). Однако нужно учитывать, что наглядность блок-схем теряется с увеличением длины алгоритма. Если блок-схема занимает больше одной тетрадной страницы, то составление ее, на наш взгляд, неэффективно.

Разберем две задачи на ветвления. При решении первой из них составляется протокол выполнения алгоритма, предназначенного для Чертежника. Разбор второй задачи демонстрирует типичный диалог между учителем и учащимися при составлении разветвляющихся алгоритмов для Вычислителя.

Задача 1. Что нарисует Чертежник, выполнив следующий алгоритм из исходного положения, показанного на рис. 4?

Если впереди край, то:
Повернуть налево.
Конец ветвления.
Если впереди край, то:
Повернуть налево.
Сделать шаг.
Повернуть налево.
Сделать шаг.
Иначе:
Сделать шаг.
Повернуть налево.
Сделать шаг.
Повернуть налево.
Сделать шаг.
Повернуть налево.
Сделать шаг.
Конец ветвления.

Решение

Приведем протокол выполнения этого алгоритма. Слева будем писать действия Чертежника, справа — комментарии к их исполнению и рисунки.

Если впереди край, то: Условие выполняется. Исполняем ветвь ветвления (ветвление — в неполной форме).
Повернуть налево.
Конец ветвления. Ветвление закончено.
Если впереди край, то: Условие выполняется. Выполняем первую ветвь ветвления (ветвление — в полной форме).
Повернуть налево.
Сделать шаг.
Повернуть налево.
Сделать шаг.
Иначе: Выполнение ветви закончено. Переходим к указателю «Конец ветвления».
Конец ветвления. Ветвление закончено.
Исполнение алгоритма завершено.

Задача 2. Даны координаты двух точек в прямоугольной системе координат. Составьте для Вычислителя алгоритм, с помощью которого он сможет определить, какая из этих точек находится дальше от начала координат.

Решение

Приведем возможный диалог, демонстрирующий разбор этой задачи.

Вопрос 1. Можно ли составить линейный алгоритм для определения ближайшей к началу координат точки?

Ответ. Нельзя, поскольку выбор той или иной точки зависит от расстояний этих точек до начала координат.

Вопрос 2. Какие условия нужно проверять?

Ответ. Пусть p — расстояние от первой точки до начала координат, а q — расстояние от второй точки до начала координат. Тогда в алгоритме надо сравнить p и q. (Здесь важно добиться, чтобы учащиеся сформулировали условие в виде сравнения чисел, ибо именно такая проверка является допустимым действием Вычислителя.)

Вопрос 3. Какие при этом возникают случаи?

Ответ. Возможны три случая: $p>q$, $p<q$ и $p=q$.

Вопрос 4. Какие действия нужно совершать при выполнении каждого из этих условий?

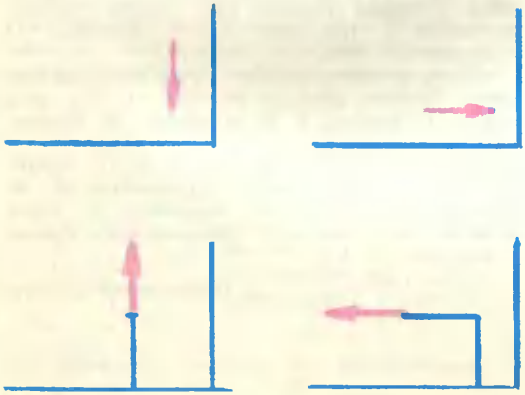
Ответ. При $p>q$ надо сообщить «Первая точка дальше»; при $p<q$ надо сообщить «Вторая точка дальше»; при $p=q$ надо сообщить «Точки удалены одинаково».

Вопрос 5. Можно ли в этом алгоритме обойтись одним ветвлением?

Ответ. Нет, поскольку одно ветвление может содержать не более двух ветвей, т. е. всегда нужно выполнять один из двух наборов действий. А в нашем случае нужно выполнять один из трех наборов действий.

Вопрос 6. Сколько ветвлений надо исполь-

4



зовать в этом алгоритме? Каковы формы этих ветвлений?

Ответ. Можно использовать три ветвления в неполной форме — по одной ветви на каждое из трех указанных выше условий.

Приведем требуемый алгоритм и одновременно протокол его выполнения.

Запросить *a, b, c, d.*
Запрашиваются значения координат первой точки (*a* и *b*) и второй точки (*c* и *d*).

Присвоить *p* **значение** $\sqrt{a^2+b^2}$.
Переменной *p* присваивается значения расстояния от начала координат до первой точки.

Присвоить *q* **значение** $\sqrt{c^2+d^2}$.
Переменной *q* присваивается значения расстояния от начала координат до второй точки.

Если $p>q$, **то:**
Проверяется условие $\sqrt{a^2+b^2} > \sqrt{c^2+d^2}$ (сравниваются расстояния от данных точек до начала координат). В случае невыполнения условия совершаются действия, записанные после строки «Конец ветвления». В случае выполнения условия совершается действие:

Сообщить «Первая точка дальше второй от начала координат».
Выдается сообщение о том, что первая точка находится дальше второй от начала координат.

Конец ветвления.
Ветвление закончено. Переходим к выполнению действий, записанных после строки «Конец ветвления».

Если $p<q$, **то:**
Проверяется условие $\sqrt{a^2+b^2} < \sqrt{c^2+d^2}$. В случае невыполнения условия совершается действие, записанное после строки «Конец ветвления». В случае выполнения условия совершается действие:

Сообщить «Вторая точка дальше первой от начала координат».

Выдается сообщение о том, что вторая точка находится дальше первой от начала координат.

Конец ветвления.
Ветвление закончено. Переходим к выполнению действий, записанных после строки «Конец ветвления».

Если $p=q$, **то:**
Проверяется условие $\sqrt{a^2+b^2} = \sqrt{c^2+d^2}$. В случае невыполнения условия выполнение алгоритма заканчивается. В случае выполнения условия совершается действие:

Сообщить «Точки равноудалены от начала координат».
Выдается сообщение о том, что от начала координат точки находятся на равном расстоянии.

Конец ветвления.
Ветвление закончено.

Замечание

В этом алгоритме можно воспользоваться и двумя ветвлениями: в неполной и полной форме. Например:

Запросить *a, b, c, d.*
Присвоить *p* **значение** $\sqrt{a^2+b^2}$.
Присвоить *q* **значение** $\sqrt{c^2+d^2}$.
Если $p>q$, **то:**
Сообщить «Первая точка дальше второй от начала координат».
Стоп.
Конец ветвления.
Если $p<q$, **то:**
Сообщить «Вторая точка дальше первой от начала координат».
Иначе:
Сообщить «Точки равноудалены от начала координат».
Конец ветвления.

Следует обратить внимание учащихся на необходимость в этом алгоритме команды «Стоп». Полезно разобрать работу алгоритма, когда команда «Стоп» отсутствует. Дело в том, что очень часто в подобных случаях ученики забывают записывать эту команду.

О методике решения задач из других тем раздела «алгоритмизация» мы расскажем в следующей статье.

Литература

1. Основы информатики и вычислительной техники / А. Г. Гейн, В. Г. Житомирский, Е. В. Линецкий и др. Свердловск: УрГУ, 1989.

2. *Болтянский В. Г.* Алгоритмизация внешняя и содержательная // Математика в школе. 1989. № 2.

3. *Гальперин П. Я.* Основные результаты исследований по проблеме «Формирование умственных действий и понятий»: Автореф. докт. дис. психол. наук. М., 1965.

4. Основы информатики и вычислительной техники: Пробный учебник для средних учебных заведений / Под ред. А. П. Ершова. М.: Просвещение, 1988.

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

6. *Гейн А. Г., Житомирский В. Г., Линецкий Е. В., Сапир М. В., Шолохович В. Ф.* Программно-методический комплекс по курсу школьной информатики // Информатика и образование. 1988. № 3.

Продолжение следует.

В. КАЙМИН, Л. КУГЕЛЬ, Е. КУЗНИЦКИЙ,
Н. УГРИНОВИЧ

Алгоритмизация и начала программирования

Основная цель изучения начал программирования в школьном курсе информатики по предлагаемой методике — освоение ЭВМ как инструмента для решения различного рода задач.

При таком подходе (рис. 1) к изучению методов составления алгоритмов и программ освоение начал программирования самым тесным образом связывается с изучением основ алгоритмизации и предполагает систему хорошо подобранных упражнений и задач для закрепления навыков отладки программ и выработки умений составления алгоритмов для решения задач на ЭВМ.

1

Для обучения началам программирования с дидактической точки зрения предпочтительнее те языки программирования, которые используют лексику родного языка, например Рапира или Е-практикум. При наличии таких языков описания алгоритмов могут производиться непосредственно на них. Для языков с иностранной лексикой типа Бейсик, Фокал или Паскаль и для обучения, и для выполнения разработок полезно предварительное описание алгоритмов на некотором языке с родной лексикой. В этих целях удобно использовать язык описания структурированных алгоритмов из учебника А. П. Ершова (который фактически выполняет роль псевдокода — языка описания логики работы программ), пополнив его операциями ввода-вывода информации и немного видоизменив запись некоторых команд, чтобы упростить технику кодирования алгоритмов по принципу «строка в строку».

Отход от блок-схем при описании алгоритмов связан со следующими причинами. При использовании структурированных алгоритмов в качестве описаний логики работы программ существенно упрощается их составление, которое превращается в достаточно простую процедуру кодирования программ по описанию алгоритмов. Во-вторых, структурированные алгоритмы и программы, как правило, намного проще читать, понимать и исправлять, чем блок-схемы. И, в-третьих, для структурированных алгоритмов и программ имеются систематические методы анализа правильности и выявления ошибок.

Приведем примеры записи операций ввода-вывода информации в алгоритмической нотации (на русском языке) и их реализацию на Бейсике (рис. 2), где x, y, $x1$, $y1$ — координаты точек на экране дисплея, c — цвет точек, r — радиус окружности, $a1$, $a2$ — углы, задающие дугу окружности.

2

```
вывод ("привет")                PRINT "привет"
запрос ("a=",a)                 INPUT "A=";A
вывод  ("b=",b)                 PRINT "B=";B
точка  (x,y),c                  PSET (X,Y),C
отрезок (x,y)-(x1,y1),c         LINE (X,Y)-(X1,Y1),C
рамка (x,y)-(x1,y1),c           LINE (X,Y)-(X1,Y1),C,B
окружность (x,y),r,c            CIRCLE (X,Y),R,C
дуга (x,y),r,c,a1,a2            CIRCLE (X,Y),R,C,A1,A2
```

Главным в предлагаемой методике является неразрывная связь между введением и использованием алгоритмов и записи

Продолжение. Начало см.: Информатика и образование. 1989. № 2, 3, 4.

соответствующих программ. При таком подходе алгоритмы служат описанием логики работы составляемых для ЭВМ программ, а программы должны быть точной реализацией составленных алгоритмов. Это позволяет строго разделять ошибки, возникшие при составлении алгоритмов, и ошибки в записи программ. Ошибки первого типа — *алгоритмические*, ошибки второго — *синтаксические*.

Поиск и исправление ошибок в алгоритмах и программах — одна из наиболее сложных проблем в программировании вообще и в обучении началам программирования в частности. Неумение находить и исправлять ошибки ведет к неприятию предмета и ощущению ложного чувства неспособности к программированию. Одним из подходов к решению проблемы предотвращения ошибок служит структуризация программ и описание логики их работы в форме структурированных алгоритмов. Алгоритмы и программы, записанные по правилам структурного программирования, намного проще понять и исправить, чем программы, составленные на Бейсике, Фортране или Фокале с неупорядоченным использованием операторов **goto**.

Другим средством для предотвращения и поиска ошибок в алгоритмах и программах является описание промежуточных и конечных результатов их выполнения. Для диалоговых ЭВМ роль таких описаний выполняют *сценарии*, включающие, как минимум, описания всех текстов и рисунков, появляющихся на экране ЭВМ.

Учебные занятия по нашей методике целиком посвящаются разбору одной задачи — от составления и обсуждения сценария до отладки программы и проведения экспериментов на ЭВМ. Для этого учебные занятия предпочтительнее строить в форме сдвоенных уроков в кабинете ВТ, где можно обсудить элементы теории и сразу же провести надлежащий практикум на ЭВМ.

При классно-кабинетной системе становится необходима блочно-урочная организация занятий: в классе излагаются элементы теории, обсуждаются сценарии и составляются алгоритмы, а затем в кабинете производятся кодирование и отладка программ, составленных в классе или дома. После наработки опыта кодирование алгоритмов учащимися может проводиться непосредственно на экране ЭВМ.

Во втором варианте работа в классе и в кабинете ВТ должна быть четко взаимосогласована. В целом на изучение основ алгоритмизации и начал программирования отводится 34 ч. Желательно использовать второе полугодие IX класса, удвоив часы информатики за счет переноса 1 ч из второго полугодия X класса. Рекомендуется следующая структура занятий по изучению начал программирования и основ алгоритмизации:

1. Основные средства, понятия и правила работы с программами на ЭВМ	4 ч
2. Средства ввода/вывода, сценарии и машинная графика на ЭВМ	4 ч
3. Ветвление и циклы в алгоритмах и программах	8 ч
4. Вспомогательные алгоритмы и подпрограммы	6 ч
5. Обработка таблиц с использованием массивов	6 ч
6. Организация диалоговых алгоритмов и программ	4 ч

При обучении началам программирования освоение средств (формирование навыков) полезно проводить параллельно с изучением понятий «алгоритм», «программа», «команда» и «присваивание». Лучше всего начинать с разъяснения понятия «команда». В числе первых демонстрируемых команд рационально рассмотреть команды вывода сообщений, выполнения вычислений и, при желании, операций над строками. Необходимо продемонстрировать команды и результаты их выполнения (рис. 3).

3

```
команды:                    результаты:
--------                    -----------
? "привет"                  привет
? 2*2                       4
? 4*ATN(1)                  3.14159
? "ма"+"ма"                 мама
```

Разъяснение понятия «присваивание» проще всего производить непосредственно в кабинете ВТ с примерами команд, в которых используются переменные (рис. 4).

4

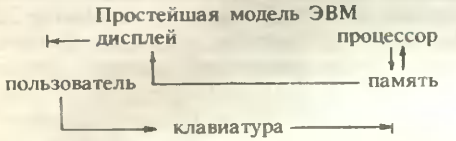
```
команды:                    результаты:
--------                    -----------
PI = 3.14159
?  PI                       3.1459
?  SIN(PI/2)                0.99997
?  COS(PI)                  0.99997
```

Для дальнейших разъяснений принципов работы ЭВМ и выполнения ими операций присваивания необходимо обсуждение простейшего представления внутренней организации вычислительных машин. На этом этапе необходимо выделить и разъяснить функции основных составных частей ЭВМ: внутренней (оперативной) памяти, процессора, устройства ввода (клавиатура) и вывода (экран ЭВМ) (рис. 5).

В рамках этой модели легко разъяснять роль ЭВМ как устройства накопления, обра-

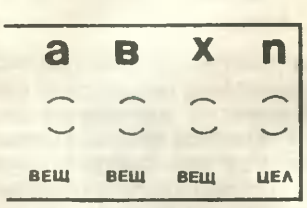
ботки и передачи данных — память служит для накопления данных, процессор — средство для их обработки, а устройства ввода-вывода — средства передачи информации. На данном этапе такая модель ЭВМ нужна для разъяснения сути понятия «переменные» в алгоритмах и программах и смысла присваиваний как машинных операций.

5



Переменные в алгоритмах и программах, в отличие от математики, это не переменные величины, а области (участки) в памяти машин (рис. 6). Суть этих «переменных» заключается в возможности хранения и обновления чисел, слов, кодов — иными словами, данных. Если в математике значение переменной может быть определено или не определено, то в алгоритмах и программах в переменных всегда хранится некоторая информация, и она может многократно обновляться по ходу выполнения команд и программ.

6



Одним из основных средств для обновления данных в машинной памяти являются машинные операции, называемые «присваивания». По форме записи присваивания похожи на равенства между величинами, используемые в математике. Но на самом деле присваивания — это операции над данными, которые должна выполнить машина, извлекая и размещая их в своей памяти. Разъяснение понятия «присваивание» проще всего провести на ЭВМ, воспользовавшись набором команд-присваиваний.

Следующим шагом в изучении начал программирования является изучение правил записи формул в алгоритмах и программах. Этот материал опять же лучше всего разбирать на примерах, предполагая, что ЭВМ помнит результаты выполнения ранее введенных команд.

Последним в изучении основных средств должен служить просмотр на ЭВМ списка математических функций, имеющихся в используемом языке программирования (см. рис. 6а).

```
команды:          функции:
--------          --------
? sin(0)     sin(x)   - синус
? cos(0)     cos(x)   - косинус
? atn(1)     arctg(x)- арктангенс
? sqr(4)     Jx       - квадратный корень
? log(1)     ln(x) - натуральный логарифм
? exp(0)     exp(x)   - экспонента
? rnd(1)              - случайное из [0:1]
```

Особо стоит обратить внимание на функцию — **rnd** (1), которая порождает «случайные числа». Это особая «функция» — ее значениями являются некоторые числа из интервала [0:1]. Причем при каждом новом обращении к этой функции она дает новое значение из интервала [0:1], что полезно проверить на ЭВМ.

На первых занятиях полезно непосредственно на ЭВМ проверить и все команды вывода графической информации, если таковые есть в используемом языке программирования. Полезно посмотреть средства вывода линий, рамок, окружностей, дуг, закраски фигур и т. п.

Отработав на ЭВМ все базовые средства обработки и вывода информации, можно переходить к обсуждению понятий «алгоритм» и «программа». Понятие программы, как последовательности команд для ЭВМ, безусловно, следует начинать с разбора примеров на машине. Например, с простейшей программы вычисления времени движения по заданной скорости и известной длине пути (рис. 7).

7

```
10 REM  время движения
20 S=900
30 V=60
40 T=S/V
50 PRINT "время=";T
60 END
```

```
алг время движения
нач
  s:=900
  v:=60
  t:=s/v
  вывод ("время =",t)
кон
```

В ходе разъяснения этого примера важно не только разобрать правила записи и нумерации строк в программах на Бейсике, но и объяснить правила ввода, редактирования и запуска программ на ЭВМ с помощью команд **run** и **list**. При этом очень важно записать и разобрать алгоритм, реализованный в этой программе. И наконец, целесообразно проиллюстрировать правила внесения изменений в программу, заменив в ней операции присваи-

вания исходных данных на операции ввода их с клавиатуры (рис. 8).

8
```
10 REM  время движения
20 INPUT "путь=";S
30 INPUT "скорость=";V
40 T=S/V
50 PRINT "время=";T
60 END
```

```
алг время движения
нач
  запрос ("путь=",s)
  запрос ("скорость=",v)
  t:=s/v
  вывод ("время =",t)
кон
```

Проработка на ЭВМ этих программ и ознакомление с соответствующими алгоритмами позволяют перейти к обсуждению понятия «алгоритм». Среди разнообразных определений этого понятия мы рекомендуем разъяснять алгоритмы не как предписания исполнителям, а как некоторые последовательности действий, которые должны выполнять некоторые устройства, машины, роботы, люди и т. п. К числу такого рода предписаний могут относиться программы для ЭВМ и станков с программным управлением, нотные записи для музыкантов, кулинарные рецепты, инструкции для сотрудников и т. п.

Отличием алгоритмов от предписаний алгоритмического типа является точность и однозначность правил их выполнения. Определяющими свойствами алгоритмов считаются однозначность выполнения каждого из составляющих его действий, конечность времени выполнения каждого из этих действий и результативность выполнения алгоритмов в целом.

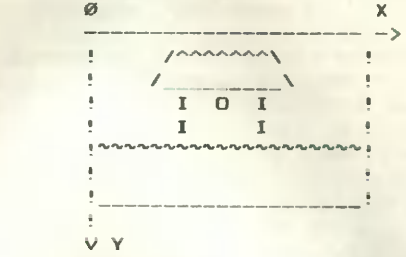
Акцент на результативность алгоритмов можно и нужно связать с результативностью деятельности вообще. Для нас в большинстве случаев важна не столько деятельность сама по себе, сколько ее результаты и их правильность. Понимание результатов выполнения составляемых алгоритмов и программ является ключевой проблемой в обучении алгоритмизации и началам программирования. Глубинный смысл этой проблемы самым тесным образом связан с умениями осознавать конечные результаты в любом виде деятельности, и в составлении программ для ЭВМ в частности.

Этим целям в обучении началам программирования служат сценарии — полные детальные описания результатов работы ЭВМ при выполнении диалоговых программ, а для описания логики работы этих программ — алгоритмы, строго соответствующие выбранным сценариям. Составление сценариев и алгоритмов в целях формирования умений осознавать конечные результаты деятельности необходимо подключить уже на самых первых простейших программах, включая затем анализ результатов выполнения промежуточных действий, а в дальнейшем — элементы пошаговой детализации алгоритмов как средства обучения планированию целенаправленной деятельности.

Проще всего понятие сценария и идеи пошаговой детализации алгоритмов ввести на задачах рисования картинок на экране ЭВМ. Одним из простейших примеров может служить задача рисования домика (рис. 9).

9

сценарий:




```
План "изображение домика"
нач
  крыша
  стена
  окно
  земля
кон
```

Первым шагом в разработке сценария является выполнение на доске или бумаге изображения домика с нанесением системы координат и отметкой на осях координат опорных точек. При этом следует указать на особенность системы координат экрана ЭВМ — ось OY направлена не вверх, как это принято в математике, а вниз. По уточнении координат точек составляется детальный алгоритм вывода изображения домика и уже затем проводится кодирование алгоритма на выбранный язык программирования (рис. 10).

10

```
алг  изображение домика
нач
  отрезок( 70, 80)-(170, 80)
  отрезок( 70, 80)-(120, 40)
  отрезок(120, 40)-(150, 40)
  отрезок(150, 40)-(170, 80)
  рамка   ( 80, 80)-(160,160)
  рамка   (120, 90)-(150,120)
  отрезок(  0,160)-(255,160)
кон
```

```
10 REM  изображение домика
[15 SCREEN 2]
20 LINE (70,80)-(170,80),2
30 LINE (70,80)-(120,40),2
40 LINE (120,40)-(150,40),2
50 LINE (150,40)-(170,80),2
60 LINE (80,80)-(160,160),2,B
```

```
70 LINE (120,90)-(150,120),2,B
80 LINE (0,160)-(255,160),2
90 END
```

Разобранный пример проверяется на ЭВМ каждым из учеников, после чего дается задание на самостоятельное составление алгоритма и программы для некоторого простого рисунка или чертежа. При этом перед проверкой программы на ЭВМ ученикам разъясняются правила ввода и отладки программ по частям. В приведенном примере программу целесообразно вводить четырьмя частями в соответствии с планом изображения домика: 1) вывод крыши, 2) вывод стены, 3) вывод окна, 4) вывод земли. Достоинством этого приема является наиболее быстрое выявление в программе всех синтаксических и алгоритмических ошибок, поскольку программа вводится по 2—3 оператора и сразу же проверяется пуском на выполнение.

Общая схема записи выбора действий по условию и ее реализация на Бейсике, отвечающая принципу кодирования «строка в строку», имеет вид (рис. 11):

11
```
если <условие1>  то
   <действия1>
инес <условие2>  то
   <действия2>
иначе
   <действия3>
кесли

100 IF NOT <условие1> GOTO 200
...   <действия1>
190 GOTO 390
200 IF NOT <условие2> GOTO 300
...   <действия2>
290 GOTO 390
300 'иначе
...   <действия3>
390 'кесли
```

В конкретных ситуациях возможно несколько альтернативных групп, начинающихся со слов «иначе если» (или их сокращения — «инес»). Возможно, что эти альтернативные группы будут отсутствовать, но общая схема кодирования сохраняет свой вид.

В качестве иллюстрации правил кодирования выбора действий приведем запись на Бейсике алгоритма решения квадратного уравнения (рис. 12).

12
```
алг квадратное уравнение
нач
  вывод ("квадратное уравнение")
  вывод ("a*x^2 + b*x + c")
  запрос ("a=",a)
  запрос ("b=",b)
  запрос ("c=",c)
  d:=b^2-4ac
  если a=0 то
    вывод ("недопустимо:a=0")
  инес d<0 то
    вывод("недопустимо:b^2-4ac<0")
  иначе
    вывод ("корни:")
    d:=√d
    x1:=(-b+d)/2a
    x2:=(-b-d)/2a
    вывод ("x1 =",x1)
    вывод ("x2 =",x2)
  кесли
кон

10 REM квадратное уравнение
20 PRINT "квадратное уравнение"
30 PRINT "A*X^2 + B*X + C = 0"
40 INPUT "A=";A
50 INPUT "B=";B
60 INPUT "C=";C
70 D=B^2-4*A*C
80 IF NOT A=0 GOTO 140
90 PRINT "недопустимо: A=0"
100 GOTO 210
110 IF NOT D<0 GOTO 140
120 PRINT "недопустимо:B^2-4AC<0"
130 GOTO 210
140 'иначе
150 PRINT "корни:"
160 D=SQR(D)
170 X1=(-B+D)/(2*A)
180 X2=(-B-D)/(2*A)
190 PRINT "X1=";X1
200 PRINT "X2=";X2
210 'кесли
220 END
```

В примере использовано наиболее общее правило кодирования структурированных алгоритмов с выбором действий на Бейсике. Бейсик, как Фортран и Фокал, отличает отсутствие полного набора средств для записи структурированных программ. Поэтому структурированные алгоритмы на этих языках приходится кодировать с помощью операторов **goto...** и **if... goto...**— условного и безусловного перехода.

Отличительной особенностью программ, составленных таким способом, является то, что по ним легко восстанавливать исходные алгоритмы. По этой причине эти программы в той же мере легко читать, понимать и исправлять, как и исходные структурированные алгоритмы. Это достоинство особенно поймут учителя и преподаватели во время диалоговой отладки программ на ЭВМ, когда в кабинете ВТ одновременно работают несколько учащихся, каждый — с собственной программой.

Ситуации, возникающие во время занятий в кабинете ВТ, часто похожи на сеанс в шахматы на нескольких досках, когда учителю необходимо в считанные секунды разобрать-

ся в причинах ошибок учащихся и найти правильный ход — очередное исправление в их программах. Поэтому завершение самостоятельной отладки программ в отведенное время становится серьезной проблемой и для ученика, и для учителя.

Для удачного завершения работ по отладке программ и решению с их помощью задач на ЭВМ необходимо, чтобы ученики обязательно представляли не только тексты программ, но и описание алгоритмов и сценариев. Большинство алгоритмических ошибок выявляется сопоставлением алгоритмов и программ со сценариями и сопоставлением результатов на экране с соответствующими фрагментами сценария. При этом принципиально важно, чтобы алгоритмы были записаны строго в соответствии с правилами структурного программирования. Если же алгоритм написан в соответствии со сценарием, а в сценарии были предусмотрены ясные и понятные тексты и сообщения при вводе-выводе данных, то поиск причин ошибок в алгоритмах и программе и их исправление по приобретении опыта завершаются в среднем за 5—10 с.

Ввод и отладку программы рекомендуется проводить по частям. В качестве иллюстрации рассмотрим упрощенные алгоритм и программу решения квадратного уравнения (рис. 12 а). В данном случае порядок ввода: 1) вывод заголовка, 2) ввод исходных данных, 3) вычисление и вывод первого корня, 4) вычисление и вывод второго корня. В этом примере главное — выявить ошибку, заложенную в алгоритм. Для этого нужно ввести данные для вырожденных случаев решения квадратного уравнения: а) отрицательный дискриминант: $a=1$, $b=1$, $c=1$ ($b^2-4ac=-3$), б) нулевой коэффициент $a=0$, $b=1$, $c=1$. И в том, и в другом случае при запуске программы должно произойти аварийное завершение. Разъяснение причин этого сбоя в работе машины потребует проведения анализа выполнения вычислений в алгоритме (рис. 13).

```
алг квадратные уравнения
нач
  вывод ("квадратные уравнения")
  вывод ("a*x^2 + b*x + c = 0")
  запрос ("a=",a)
  запрос ("b=",b)
  запрос ("c=",c)
  d:=√(b²-4ac)
  x1:=(-b+d)/2a
  x2:=(-b-d)/2a
  вывод ("корни:")
  вывод ("x1=",x1)
  вывод ("x2=",x2)
кон
```

```
10 REM квадратные уравнения
20 PRINT "квадратные уравнения"
30 PRINT "A*X^2 + B*X + C = 0"
40 INPUT "A=";A
50 INPUT "B=";B
60 INPUT "C=";C
70 D=SQR(B^2-4*A*C)
80 X1=(-B+D)/2*A
90 X2=(-B-D)/2*A
100 PRINT "корни:"
110 PRINT "X1=";X1
120 PRINT "X2=",X2
130 END
```

13

```
алгоритм          результаты
--------          ----------
d :=√(b²-4ac)     { d =√(b²-4ac)
x1:=(-b+d)/2a     { x1=(-b+d)/2a
x2:=(-b-d)/2a     { x2=(-b-d)/2a
```

В приведенном примере необходимы замечания по форме записи. Слева приведены записи трех присваиваний, а справа — три соотношения между вычисляемыми и исходными величинами. Если алгоритмическая запись используется для описания правил работы ЭВМ, то математическая запись необходима для анализа результатов выполнения алгоритма. В данном случае из приведенных соотношений между величинами x^1, x^2, d, с одной стороны, и величинами a, b, c, с другой стороны, мы можем вывести соотношения между исходными данными и конечными результатами вычислений — известные формулы вычисления корней квадратных уравнений, которые могут применяться только при $a\neq 0$ и $b^2-4ac\geqslant 0$. Поэтому мы можем заявить, что алгоритм правильно вычислит корни при допустимых значениях коэффициентов a, b, c, но даст сбои при недопустимых данных.

Устранение ошибки в этом алгоритме и программе требует их переработки с учетом результатов их работы при возможных недопустимых комбинациях исходных данных. Данную проблему прежде всего следует решать, составляя сценарий работы ЭВМ, а затем предусматривая «ветвления» в алгоритме. Приведем пример «надежного» варианта алгоритма решения квадратного уравнения, в котором отвергаются недопустимые данные (рис. 14).

14

```
алг квадратное уравнение
  нач
    вывод ("квадратное уравнение")
    вывод ("a*x^2 + b*x + c = 0")
    запрос ("a=",a)
    запрос ("b=",b)
    запрос ("c=",c)
    d:=b^2-4ac
    если  a=0 то
      вывод("недопустимо:a=0")
    инес d<0 то
```

```
    вывод("недопустимо:b^2-4ac<0")
  иначе
    d:=√d
    вывод ("корни:")
    x1:=(-b+d)/2a
    x2:=(-b-d)/2a
    вывод ("x1=",x1)
    вывод ("x2=",x2)
  кесли
кон
```

Сценарий

```
  квадратное уравнение
  a*x^2 + b*x + c = 0
  a = ? <<a>>
  b = ? <<b>>
  c = ? <<c>>

{ недопустимо: a=0

{ недопустимо: b^2-4ac<0

{ корни:
{ x1= <x1>
{ x2= <x2>
 где { x1=(-b-d)/2a
     { x2=(-b+d)/2a
     { d=b^2-4ac
```

Здесь приведен сценарий, в котором указаны все виды текстов и сообщений, которые могут появляться на экране ЭВМ при выполнении составленной программы. Выше приведен соответствующий алгорим, в котором предусмотрен выбор различных групп действий. Если коэффициент a окажется равным 0, то алгоритм предусматривает вывод сообщения «недопустимо: $a=0$» — и только его. Если же выявится, что отрицательна величина дискриминанта $d=b^2-4ac$, то будет выводиться сообщение «недопустимо $b^2-4ac<0$» — и только оно. И наконец, если выполнятся условия $a\neq 0$ и $d\geqslant 0$, то алгоритм предусматривае вычисление и вывод корней. Такой спосбб группирования действий в алгоритмах мы называем выбором действий по условиям. Это одно из основных правил структурного проектирования в составлении алгоритмов и программ.

В то же время необходимо обратить внимание, что для кодирования условий выбора «если...то...» на Бейсике приходится использовать правило «антиподов» — слово «если...» в программе на Бейсике приходится кодировать наоборот — «**if not...**» (рис. 15). Переводы этих фраз с Бейсика на русский звучат примерно так: «если неверно, что $a=b$, то идти...», «если неверно, что $(0<a)$ и $(a<1)$, то идти...» и т. п.

15

```
    алгоритм
    --------
если a=b то ...

если (0<a) и (a<1) то ...
```

Бейсик

```
IF NOT (A=B) GOTO ...

IF NOT ((0<A) AND (A<1)) GOTO ...
```

Сложности в поиске и исправлении ошибок могут резко возрасти при нарушении правил структурного программирования и неконтролируемом использовании операторов **goto**. Эти трудности исчезают при использовании для обучения началам программирования языков типа Паскаль или Рапира. Не случайно на всех олимпиадах по информатике и программированию, как правило, побеждают те школьники и студенты, которые предпочитают составлять программы на языке Паскаль, а учащиеся, предпочитающие Бейсик, Фокал или Фортран, не успевают завершить отладку своих программ.

Второй типичной причиной проникновения ошибок в программы являются неудачные попытки «оптимизировать» или упростить программы за счет использования дополнительных возможностей Бейсика. Например, использование условного оператора **if... then...** для реализации выбора действий вида (рис. 16):

16

```
если <условие> то
   <действие1>
  иначе
   <действие2>
кесли
```

```
IF  <условие>
 THEN <оператор1>
 ELSE <оператор2>
```

При использовании этого правила следует помнить, что оператор **if... the... else...** правильно реализует только выбор, если «действия» можно записать в виде единственного оператора. В противном случае возможны различные тонкие ошибки. Точно такого же рода тонкие ошибки могут проникнуть в программу при использовании упрощенного варианта кодирования условных действий в виде условного оператора на Бейсике (рис. 17). Здесь нужно помнить, что все операторы должны быть записаны в одну строку, а длина этой строки не должна превышать 256 символов.

17

```
если <условие> то
   <действия1>
кесли
```

```
IF  <условие>
 THEN <операторы1>
```

Причиной ошибок третьего рода может быть неправильная запись отрицаний условий в операторах «**if not...**», когда в целях упрощения отрицание **not** заменяют на отрицание исходного условия.

Разбор дальнейших примеров алгоритмов и программ с выбором действий следует пред-

18

Условия	Бейсик
a=b	A=B
a≠b	A<>B
a<b	A<B
a≤b	A<=B
a>b	A>B
a≥b	A>=B
не (a>0)	NOT (A>0)
(0<a) и (a<1)	(0<A) AND (A<1)
(a=0) или (a=1)	(A=0) OR (A=1)

```
NOT (A=B)   -   A<>B
NOT (A<>B)  -   A=B
NOT (A>B)   -   A<=B
NOT (A>=B)  -   A<B
NOT (A<B)   -   A>=B
NOT (A<=B)  -   A>B
NOT ((0<A) AND (A<1))=(A<=0) OR (1<=A)
NOT ((A=0) OR (A=1)) =(A<>0) AND (A<>1)
```

варить разбором общих правил кодирования простых и составных условий на Бейсике. Для этого хорошо иметь плакаты (рис. 18).

Правила, представленные на плакатах, следует разбирать с учащимися, только если они ранее были изучены в терминах «и», «или», «не» при изучении элементов логики. В таком случае они закрепляют навыки доказательных рассуждений, необходимых для анализа правильности алгоритмов и программ и выявления в них ошибок.

Продолжение следует.

А. АВЕРБУХ, В. ГИСИН, Я. ЗАЙДЕЛЬМАН

Методические рекомендации к новому учебнику информатики

Конспект § 1—8 учебника и распределение материала по урокам опубликованы в предыдущем номере журнала. В данной статье излагается конспект § 3—8 методического пособия, которое планируется к изданию в 1990 г. Содержание и методика преподавания § 1—2 (Предмет информатики. ЭВМ — универсальная машина для обработки информации) традиционны. Их изложение в журнальном варианте опущено. Методические рекомендации к каждому параграфу пособия состоят из пяти частей:

1) цели и учебные задачи, требования к знаниям и умениям учащихся;

2) основные понятия;

3) общие указания к параграфу;

4) подробные рекомендации по ведению уроков;

5) методические указания к решению задач, разбор возможных вариантов, усложнений и т. п.

Задачи и упражнения в учебнике просты и доступны любому ученику. Анализ решений, приведенный в методическом пособии, раскрывает развивающий потенциал задач, позволяет учителю варьировать их сложность.

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

§ 3. Алгоритмы управления Роботом (2 ч)

В параграфе изучается первый из четырех исполнителей — Робот. Изучение такого исполнителя удобно тем, что элементарные действия Робота просты, наглядны, легко могут быть выполнены в тетради.

Учитель должен ясно представлять, на овладение какими понятиями нацелены внешне очень простые примеры учебника.

Исполнитель. Слово «исполнитель» в учебнике практически не встречается. Точное определение исполнителя дать трудно, да в этом и нет необходимости. Важно понять, чем характеризуется исполнитель: *среда, система команд, элементарное действие, отказы.*

Среда или обстановка — «место обитания» исполнителя. Например, среда Робота — бесконечное клетчатое поле, ориентированное по сторонам горизонта, стены и закрашенные клетки на поле.

Система команд исполнителя (СКИ). Каждый исполнитель может выполнять команды только из некоторого конечного списка — системы команд исполнителя. Для каждой команды должны быть описаны условия применимости и результаты ее выполнения.

Исполнителя можно представить в виде устройства с кнопочным управлением. Каждая кнопка соответствует одной команде исполнителя, ее нажатие означает вызов этой команды.

Управление исполнителями заключается в последовательном вызове команд. В простейшем случае можно считать, что исполнителем управляет человек. Однако человек может ошибиться при наборе длинной последовательности команд, неверно проанализировать обстановку, не успеть в критической по времени ситуации. В дальнейшем предполагается, что управление исполнителями происходит с помощью ЭВМ.

В этом случае управление распадается на несколько этапов (рис. 1):

Содержание этапов:

1. Составление алгоритма.

Человек анализирует поставленную задачу, составляет алгоритм и передает его ЭВМ.

1

2. Исполнение алгоритма.

ЭВМ, выполняя алгоритм, управляет исполнителем, при необходимости получая от него информацию. Часть времени ЭВМ расходуется на обработку информации без обращения к исполнителю.

3. Получение результатов.

Результатами исполнения алгоритма пользуется человек. Этими результатами может быть как работа, выполненная исполнителем, так и информация, полученная от ЭВМ.

На ЭВМ полностью распространяется *принцип формального исполнения:* ЭВМ не анализирует содержание и смысл алгоритма, она последовательно, команда за командой, исполняет его. Отсюда следует необходимость точной, не допускающей двусмысленности формы записи алгоритма. Такую форму обеспечивают формальные языки, один из которых — школьный алгоритмический язык — описан в учебнике.

Несколько замечаний о понятии алгоритма. Это понятие и основные свойства алгоритмов рассматриваются на примерах в предыдущих параграфах. Здесь отмечаются те особенности, с которыми учащиеся столкнутся при изучении § 3.

1. Одна и та же последовательность действий исполнителя может быть записана разными способами, как на одном алгоритмическом языке, так и на разных алгоритмических языках.

2. Можно провести аналогию между выполнением алгоритма и выполнением одной команды исполнителя. В обоих случаях есть ограничения на применение, есть результат, можно не знать механизм выполнения действий. По существу, готовые алгоритмы ничем не отличаются от команд исполнителя. Эта мысль развивается при изучении вспомогательных алгоритмов, но показать ученикам сходство команд и алгоритмов, обратив внимание на роль заголовка, полезно в самом начале изучения алгоритмического языка.

3. Алгоритм дает решение не одной, а целого класса задач, называемого *областью применимости*. В случае Робота (применительно к материалу § 3) каждая конкретная задача определяется исходным положением Робота. Например, задача выполнить «ход конем» может быть поставлена для любой клетки поля Робота. Алгоритм дает решение всех таких задач.

Для многих задач, например для любого из заданий в задаче 5*, можно указать

---

* Нумерация параграфов учебника и задач приводится в соответствии с предыдущей статьей.— *Примеч. ред.*

бесконечное число алгоритмов. Поэтому полезно рассмотреть алгоритмы, удовлетворяющие некоторым дополнительным требованиям. Например: алгоритм содержит минимальное число команд, при исполнении алгоритма Робот не проходит дважды ни через одну клетку и т. п. Затем можно проанализировать, когда эти требования определяют алгоритм однозначно.

При изучении § 3 на уроках прежде всего вводится понятие системы команд и среды исполнителя. Например, система команд автоматического устройства из задачи 2 включает две команды: 1) удвоить; 2) увеличить на единицу.

Задание. Составьте описание системы команд и среды исполнителя Крестьянин из задачи 1.

Решение

Система команд	Среда
влево (коза)	река
влево (волк)	левый берег
влево (капуста)	правый берег
влево (ничего)	лодка
вправо (коза)	волк
вправо (волк)	коза
вправо (капуста)	капуста
вправо (ничего)	

Далее изучается исполнитель Робот с пятью командами (перемещения и закраски). Рассматривается управление исполнителем с помощью ЭВМ, составляются простые алгоритмы управления Роботом. Исполнение ролей ЭВМ и Робота при анализе алгоритмов на доске можно поручить двум ученикам.

Затем имеет смысл подробно рассмотреть вопрос об ошибках в алгоритмах. Различают три типа ошибок (табл. 1, для примера взята задача 1).

Каждый ученик в конце тетради может вести **словарь служебных слов.** При повторении и закреплении материала запишем:

алг *заголовок алгоритма*

дано *условие, проверяемое перед исполнением алгоритма*

надо *условие, проверяемое после исполнения алгоритма*

Таблица 1

№ п/п	Смысл ошибки	Название	Пример
1	Цель не достигнута	Логическая	Волк съел козу
2	Исполнитель выходит из строя (отказ)	Семантическая	Коза на левом берегу, дана команда: влево (коза)
3	Команда не входит в СКИ исполнителя	Синтаксическая	Дана команда: суши весла!

нач *начало тела алгоритма*

кон *конец алгоритма*

вертикальная черта *комментарий* (в журнальном варианте — восклицательный знак)

§ 4. Чертежник и работа с ним (2 ч)

Важнейшее новое понятие этого параграфа — **команды с аргументами.** При вызове некоторых команд недостаточно сообщить исполнителю, *что* он должен сделать. Необходимо передать ему некоторую *дополнительную информацию.* Эта информация передается с помощью аргументов.

В § 4 изучаются два новых исполнителя: Чертежник и Черепашка. Элементарные действия Чертежника достаточно просты. В то же время освоение этих действий требует от учащихся умения свободно оперировать с декартовыми координатами точек на плоскости, выполнять и записывать в координатах сложение векторов. Поэтому изучение Чертежника целесообразно сочетать с повторением темы «Векторы на плоскости». Материал параграфа дает возможность в широких пределах варьировать математическую трудность заданий.

Команда Чертежника «сместиться в точку» проще для освоения, чем команда «сместиться на вектор». Однако последняя играет важную роль при составлении вспомогательных алгоритмов. Алгоритм для Чертежника, составленный из команд «сместиться на вектор», позволяет с помощью одного и того же алгоритма получать изображение в разных местах плоскости, меняя лишь начальную точку. При изучении команды «сместиться на вектор» нужно напомнить операцию сложения векторов: если перо находится в точке A, то после исполнения команды «сместиться на вектор (dx, dy)» перо сместится в точку B такую, что $\mathrm{OB}=\mathrm{OA}+(\mathrm{dx}, \mathrm{dy})$; последовательное выполнение нескольких команд «сместиться на вектор» приводит к перемещению пера на суммарный вектор.

В задачах можно предложить несколько усложнений. Приведем два примера для задачи 8б.

1. Указать общий вид серии команд Чертежника, при исполнении которой будет нарисован отрезок, параллельный оси OX и удовлетворяющий условиям задачи (ответ: сместиться в точку $(x, 2)$; опустить перо; сместиться на вектор $(3,0)$, где $-1 \leqslant x \leqslant 2$.

2. Нарисовать требуемый отрезок так, чтобы он не был параллелен осям координат (в решении используется команда «сместиться на вектор (a, b), где a, b — конечные десятичные дроби, такие, что $a^2+b^2=3$; например (1.8, 2.4), (0.74, 2.98). Приведенные решения можно получить подбором. Поиск

общего метода связан с решением интересных математических задач.

При изучении § 4 на уроке можно начать с задачи 8а.

алг отрезок

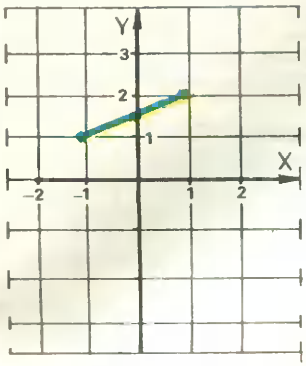
дано ! перо в исходном положении
надо ! нарисован отрезок (рис. 2)

нач

- сместиться в точку (1,2)
- опустить перо
- сместиться в точку (—1,1)
- поднять перо

кон

2



Вопрос: «Что будет, если выполнить команды алгоритма «отрезок» в обратном порядке?»

Ответ: «Будет нарисован тот же самый отрезок, только перо останется опущенным и будет находиться в точке (1, 2)».

Затем вспомним задачу о волке, козе и капусте (задача 1). Команда переправить козу на правый берег выглядела так: вправо (коза). В командах такого типа есть аргумент (в данном случае — коза). В общем виде систему команд Крестьянина можно коротко представить так (сравните с изложенным ранее):

влево (x)
вправо (x),

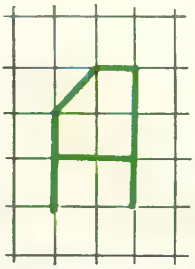
где x= волк, коза, капуста, ничего.

У Чертежника мы тоже встречаем команды с аргументами.

Вопрос: «Что будет нарисовано при исполнении алгоритма «отрезок», если поменять знаки всех аргументов на противоположные?»

Ответ: «Получится отрезок, симметричный относительно начала координат, т. е. точки (0,0)».

С помощью Чертежника можно рисовать буквы и цифры из отрезков прямых. В учебнике приведен пример рисования буквы «М». Задание: составьте алгоритм рисования буквы «А» (рис. 3).



Запишем в словарь:
; *разделитель команд в строке*

§ 5. Вспомогательные алгоритмы (6 ч)

Свойство быть вспомогательным алгоритмом определяется ролью алгоритма в решении задачи, а не его собственными качествами, так же, например, как свойство быть слагаемым определяется ролью числа в выражении, а не его величиной.

Умение выделять в задаче подзадачи необходимо не только в информатике, но и во многих других дисциплинах. Структуризация задачи — одна из основных линий курса информатики. Выработка соответствующих умений и стиля мышления — одна из конечных целей курса.

Важную роль в формировании алгоритмического мышления играет *построение алгоритмов с использованием вспомогательных*. Можно выделить две стратегии такого построения, соответствующие двум технологиям программирования и двум компонентам алгоритмического мышления (табл. 2).

Таблица 2

Построение алгоритмов	Алгоритмическое мышление	Технология программирования
Использование готовых вспомогательных алгоритмов	Сведение нерешенной задачи к уже решенным	Снизу вверх
Проектирование вспомогательных алгоритмов	Разбиение большой задачи на малые	Сверху вниз

Важное понятие — *алгоритмы с аргументами*. Полное понимание механизма аргументов возможно только после изучения понятия величины (§ 11); здесь же отметим, что алгоритмы с аргументами аналогичны командам исполнителей с аргументами.

В начале урока составим алгоритм, по которому Чертежник напишет слово «МАМА». Решить эту задачу можно так:

```
алг нарисовать мама
  дано ! перо в исходном положении
  надо ! написано слово «мама», перо
       ! в начале следующей буквы
нач
· нарисовать М
· нарисовать А
· нарисовать М
· нарисовать А
кон
```

Приведем другое решение:

```
алг нарисовать мама 1
  дано ! перо в исходном положении
  надо ! написано слово «мама», перо
       ! в начале следующей буквы
нач
· нарисовать МА
· нарисовать МА
кон
алг нарисовать МА
  дано ! перо в начале первой буквы
  надо ! написано «МА», перо
       ! в начале следующей буквы
нач
· нарисовать М
· нарисовать А
кон
```

Вспомогательные алгоритмы «нарисовать М» и «нарисовать А» были составлены раньше (первый — в учебнике, второй — на предыдущих уроках).

При использовании вспомогательных алгоритмов нас не интересует, каким образом устроен вспомогательный алгоритм. Важно лишь, чтобы он получил правильные исходные данные (дано) и выдал требуемый результат (надо).

Рассмотрим задачу 18 в. Видно, что движение Робота состоит из повторяющихся участков: обход стены и три шага на восток. Составим алгоритмы, управляющие движением Робота на этих участках, а затем — основной алгоритм.

Разберем на характерном примере метод последовательного уточнения. Требуется с помощью Робота закрасить в виде шахматной доски часть поля размером 8×8 клеток. Разобьем задачу на «подзадачи». Выделим на поле четыре одинаковые полосы шириной в две клетки, идущие с запада на восток. «Подзадачу» закрашивания полосы разделим на части, выделив четыре одинаковых квадрата 2×2 клетки (полученные таким образом алгоритмы в журнальном варианте не приводятся).

Применим метод последовательного уточнения для другой задачи. Пусть требуется изобразить домик на краю елового леса (рис. 4).

Основной алгоритм может выглядеть так:

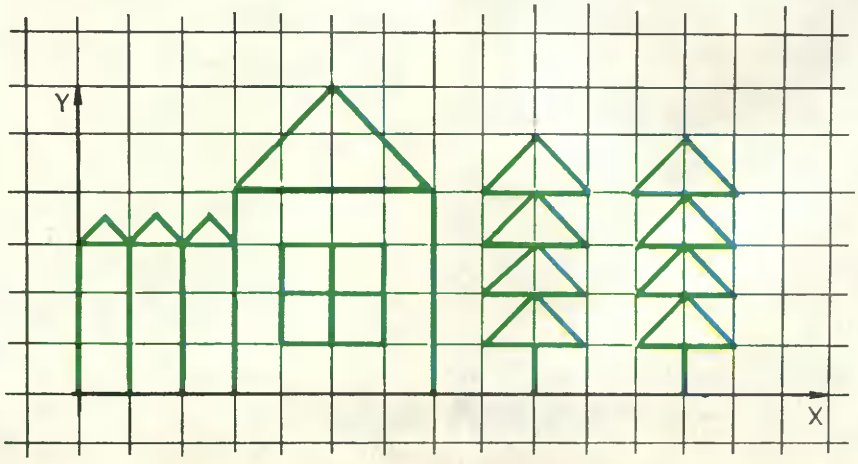
```
алг домик возле леса
  дано ! перо в исходном положении
  надо ! сделан рисунок (рис. 4)
нач
· нарисовать забор
· сместиться в точку (3,0)
· нарисовать домик с окошком
· сместиться в точку (9,0)
· нарисовать елку
· сместиться в точку (12,0)
· нарисовать елку
кон
```

Для экономии времени урока класс делится на бригады, каждая из которых разрабатывает один из вспомогательных алгоритмов. В журнальном варианте тексты вспомогательных алгоритмов для этой и других задач не приводятся.

При составлении и анализе алгоритмов с аргументами можно познакомить учащихся с понятием величины в ЭВМ (вводится в § 11).

Для тех, кто хорошо усвоил понятие исполнителя, может быть полезным такое сравнение. В состав компьютера входит исполнитель Память, который умеет записывать

4



информацию, передавать ее другим устройствам и стирать ее. Информация при этом бывает двух видов: алгоритмы и величины. При анализе работы ЭВМ по какому-нибудь алгоритму один из учеников может исполнить у доски роль Памяти.

Запишем в словарь:

арг *объявление величин-аргументов*

вещ *описание величины вещественного типа*

цел *описание величины целого типа*

§ 6. Арифметические выражения и правила их записи (1 ч)

Понятие арифметического выражения в алгоритмическом языке имеет некоторые особенности, в частности запись арифметического выражения может содержать не только знаки арифметических операций, но и обращения к функциям.

Как показывает опыт преподавания, учащиеся легко усваивают способ записи формул на алгоритмическом языке, без труда переводят выражения из традиционной формы в линейную и обратно. Сильным ученикам становится скучно. Работа алгоритмов с аргументами — существенно более сложный материал. Его можно использовать на уроке при работе с сильными учениками.

§ 7. Команды алгоритмического языка. Команда повторения n раз (2 ч)

Составные команды (повторения и ветвления) играют в алгоритмическом языке особую роль. Они не задают конкретные действия исполнителю, как, например, команды вызова. Выполняя эти команды, ЭВМ определяет последовательность вызова входящих в них простых команд. Таким образом обеспечивается возможность многократного безошибочного повторения серий команд, выбора нужной команды в зависимости от условий. Именно эти возможности лежат в основе массовых применений ЭВМ.

Ввести команду повторения на уроке можно двумя способами. Во-первых, используя свойство этой команды сокращать и упрощать запись алгоритма (так сделано в учебнике). Во-вторых, можно поставить задачу, при решении которой необходимо варьировать число повторений. Пусть, например, требуется с помощью Робота закрасить полосу длиной в четыре клетки и шириной в одну клетку. Алгоритм, не содержащий команды цикла, будет иметь узкую область применимости. Например, для полосы длиной пять клеток придется писать новый алгоритм. Как расширить область применимости этого алгоритма? Только команда цикла позволяет составить алгоритм с переменным числом повторений в качестве аргумента.

Запишем в словарь:

нц число повторений **раз** } *цикл с известным числом повторений*

· серия команд

кц

§ 8. Команда повторения для (2 ч)

Основное свойство команды **для** то же, что и у команды **раз** — количество повторений цикла известно в момент начала его исполнения.

Рассмотрим основное отличие этих циклов. В цикле *n* **раз** при каждом повторении используется одна и та же серия команд, номер повторения при исполнении цикла не используется. В цикле **для** появляется дополнительная величина — параметр цикла. Таким образом, серия команд может зависеть от номера повторения. Это различие является ключевым при решении вопроса о применении того или иного вида цикла. Если повторяются одинаковые действия, можно применить цикл *n* **раз**; если действия необходимо варьировать, нужен цикл **для**.

В учебнике наложены довольно жесткие ограничения на цикл **для**: нет шага, параметр может быть только целой величиной. Эти ограничения призваны облегчить усвоение материала учениками.

Пояснение к алгоритму «график». В нем содержится выражение вида «$f(x)$». Оно может быть вызовом алгоритма-функции. Но поскольку ученики этого пока не знают, можно считать, что это — схема алгоритма, в которую надо подставлять конкретную функцию. После изучения § 12 алгоритм построения графика функции может рассматриваться как правильно составленный алгоритм и может быть исполнен при условии, что задан вспомогательный алгоритм- функция «*f*».

Задачи параграфа позволяют учителю варьировать их математическую сложность. Например, в задаче 21а можно предложить учащимся определить точку, в которой будет находиться перо после исполнения алгоритма, не выполняя рисунка. В результате исполнения команд из тела цикла перо Чертежника смещается на вектор $(k,0)+(0,-k)+(-k-1,0)+(0,k+1)=(-1,1)$. Таким образом после трехкратного выполнения цикла перо сместится на вектор $(-3,3)$. Поскольку начальное положение пера — точка $(0,0)$, после исполнения алгоритма перо находится в точке $(-3,3)$.

На уроке после введения команды цикла с параметром полезно выполнить задания для проверки усвоения материала. Например: некоторая величина *z* принимает цело-

численные значения. Выберите из приведенного списка правильный заголовок цикла и ответьте, сколько раз будет выполнен цикл с этим заголовком при z=3:

1) нц для *i* от 0 до 0.7*z+0.33;
2) нц для *i* от *sqrt* (*z*) до *sin* (*z*);
3) нц для *i* от 1 до (2z+1)*(2z—1);
4) нц для *i* от *abs* (2*z—8) до z** 2;
5) нц для *i* от 3.14 до *z*?

Ответ: «Правильный заголовок цикла в строке 4; при z=3 этот цикл выполнится 8 раз».

Опыт преподавания по другим учебникам информатики показал, что в середине курса возникает отчуждение от предмета учеников со слабой математической подготовкой. Это было следствием возрастающей сложности материала. При работе по методике рассматриваемого учебника наблюдается обратное: к концу первого полугодия могут заскучать некоторые сильные ученики.

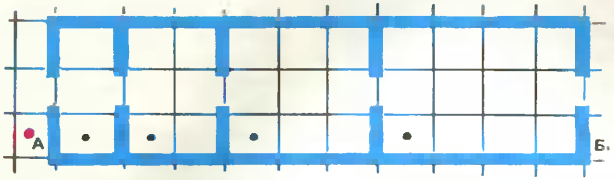
Избежать этого можно, предлагая им задачи с усложнением условий, предъявляя к алгоритмам дополнительные требования (экономия памяти, сокращение числа вызовов команд и т. п.), анализируя по шагам выполнение алгоритмов на ЭВМ, а также наполняя абстрактные задачи конкретным содержанием.

Приведем пример подобной задачи. На планете Ник созвездия Хру суровые климатические условия. Поэтому строительство ведется автоматическими устройствами с дистанционным управлением. Исполнители Закладчик Фундамента и Установщик Стен выполнили свою работу. Теперь необходимо разметить пол возле оконных проемов для последующей работы Врезчика Окон. (По сложившейся на Нике традиции, крышу дома устанавливают сами жильцы в процессе устранения отдельных недоделок). Для разметки будем использовать нашего Робота. Следует отметить некоторые особенности никских домов:

1. Дома ориентированы с запада на восток.
2. Ширина дома равна одной комнате.
3. Длина первой комнаты равна одному шагу Робота.
4. Длина каждой следующей комнаты на единицу больше длины предыдущей.
5. Окно располагается в юго-западном углу каждой комнаты.

На рис. 5 изображен никский дом из четырех комнат. Отмечены клетки, которые надо закрасить для последующей работы Врезчика Окон.

5




```
алг разметить оконные проемы
   дано ! Робот в клетке А (рис. 5)
   надо ! Робот в клетке Б
        ! нужные клетки закрашены.
нач цел i
· нц для i от 0 до 3
  · · обход
  · · закрасить
  · · нц i раз
  · · · шаг на восток
  · · кц
  · кц
  · обход
кон
```

Запишем в словарь:

```
нц для i от n1 до n2 } команда цикла
· серия команд       } с параметром
кц
```

В следующем номере предполагается опубликовать конспект § 9—12 учебника, а затем методические рекомендации к этим параграфам.

Продолжение следует.

Путешествие по Солнечной системе

ЧТО МОЖЕТ ЭВМ?

сможет совершить каждый, кто приобретет видеодиск под названием «Галерея Вояджера». На оптическом диске в цифровой форме закодировано около 3 тыс. цветных фотографий, диаграмм, карт планет и Солнечной системы, текстовых сообщений и другой информации, которую можно воспроизвести на экране телевизора.

Кроме того, на этом же диске содержится тридцать пять видеоклипов, которые дают его владельцу возможность увидеть через «иллюминатор» своего телевизора Юпитер, Сатурн, Уран и их спутники и кольца в том великолепии, в котором их мог бы наблюдать астронавт с борта космических аппаратов «Вояджер-1» и «Вояджер-2».

Видеодиск «Галерея Вояджера» выпускается как в учебном, так и в развлекательном варианте.

А. ДУВАНОВ, Я. ЗАЙДЕЛЬМАН, Ю. ПЕРВИН, М. ГОЛЬЦМАН

Роботландия — курс информатики для младших школьников

Сейчас информатика изучается в средней школе в IX—X классах. Роботландия же ориентирована на младших школьников (III—IV класс), причем это не введение, не кружковые разработки, а полный курс информатики [1, 2, 3]. Чем объясняется стремление столь сильно понизить возраст обучаемых?

Во-первых, очень важна роль курса информатики в формировании операционного стиля мышления, который рассматривается здесь в виде совокупности таких фундаментальных навыков и умений, как планирование структуры действий, поиск информации, структурирование общения, построение информационных моделей, инструментирование деятельности [4]. А ведь у старшеклассников стиль мышления уже фактически сложился, новые формы они воспринимают с трудом.

Во-вторых, курс должен научить школьников практической работе с информацией на ЭВМ. Освоив компьютер в младших классах, дети смогут затем использовать его как инструмент своей деятельности, у них не вызовут затруднений предметные уроки с применением ЭВМ. Таким же образом в старших классах дети будут иметь возможность использовать и другие полученные знания (методы работы с информацией, способы проектирования решения задач, планирование проверки полученного решения и т. д.) при прохождении различных курсов, превращая их в умения и прочные навыки. Информатика как бы растворится во множестве своих приложений.

В-третьих, компьютер позволяет превратить курс информатики в интересную игру — путешествие по стране роботов. Такой подход часто вызывает скептические улыбки у старшеклассников, но игра на уроках у младших школьников вдохновляет их, на урок приходит радость и тогда можно преодолеть даже очень трудные методические барьеры.

Имеющееся мнение [5], что младшие школьники не готовы к восприятию информатики, потому что имеют мало математических знаний, представляется неправильным. Информатика не сводится к обработке только числовой информации. Информационные процессы столь всеобщи, что сопровождают любую деятельность человека. Содержательных примеров в жизни младшего школьника вполне достаточно для построения курса информатики на их основе. Об этом свидетельствует наш собственный практический опыт, а также опыт других исследователей — советских [6], болгарских [7], американских [8].

При построении курса были поставлены следующие цели изучения информатики в школе:

формирование в сознании школьника информационной картины мира (эта задача ставит информатику в ряд естественных наук);

формирование компьютерной интуиции: знание возможностей и ограничений использования ЭВМ как инструмента для деятельности; умение использовать ЭВМ на практике только в тех случаях, когда это эффективно;

формирование операционного стиля мышления: умение формализовать задачу; выделить в ней логически самостоятельные части; определить взаимосвязи этих частей; спроектировать решение при помощи нисходящей и восходящей технологий; верифицировать результат (операционный стиль характерен для самых различных видов деятельности, а не только для программирования, как это иногда однобоко понимается);

формирование конструкторских и исследовательских навыков активного творчества с использованием современных технологий, которые обеспечивает компьютер.

Сформулированные задачи определили четыре методические линии курса, которые можно кратко обозначить ключевыми словами: информация, компьютер, программирование, творчество.

Это линии проходят через все темы курса. Каждая из них развивается по своей собственной логике, но при этом они пересекаются, поддерживают и дополняют друг друга.

В таблице сведены основные темы курса, прослеживается их участие в формировании тех или иных навыков.

На первых занятиях линии информации, компьютера и программирования идут параллельно, почти не взаимодействуя друг с другом.

Линия информации на этом этапе представлена беседами об информации, вводящими ученика в круг основных понятий информатики. Все изучаемые в дальнейшем

теоретические понятия, все практические навыки работы с компьютером будут рассматриваться и анализироваться как информационные процессы.

О чем же эти беседы? Об информации, ее фундаментальной сущности и многообразии форм. Об информационных процессах: передаче, хранении, обработке информации. Все это довольно сложные понятия, их абстрактная сущность не до конца понятна младшим школьникам, поэтому важно добиться не заучивания формальных определений (их в курсе почти нет), а практического освоения этих понятий. Третьеклассни-

Основные темы курса информатики Роботландия

Тема	Информация	Компьютер	Программирование	Творчество
1. Введение. Беседы об использовании ЭВМ		Первичные представления		
2. Беседы об информации	Распознавание, передача, хранение, обработка	ЭВМ — инструмент для работы с информацией		
3. Работа за клавиатурой	Преобразование информации в ЭВМ	Первичные навыки		
4. Алгоритмические этюды	Преобразование информации	Клавиатура, модели на ЭВМ	Понятие алгоритма, формализация записи	Логическое мышление
5. Устройство компьютера	Передача, хранение, обработка информации внутри ЭВМ	Назначение и взаимосвязь частей ЭВМ		
6. Исполнители	Информационные процессы в исполнителях	ЭВМ как универсальный исполнитель. Моделирование исполнителя	Исполнитель, система команд, синтаксические и семантические ошибки, формальное исполнение	
7. Черные ящики	Многообразие форм информации и способов ее обработки	Автоматизированное рабочее место исследователя за ЭВМ	Алгоритмы обработки информации	Навыки самостоятельного исследования, интуиция, логические выводы
8. Редакторы	Многообразие форм информации: рисунки, тексты, звук.	Инструментальные применения ЭВМ. Понятие редактирования	Редакторы как исполнители	Конструирование, сочинение
9. Технология творчества*				Формирование навыков логического мышления
10. Сложные исполнители	Обработка информации	Сложные системы: редактор текста + + редактор среды + исполнитель	Язык, программа, управляющие структуры, проектирование	Логическое мышление, конструирование, верификация
11. Банки данных*	Создание, поиск, обработка, хранение информации	ЭВМ — инструмент для работы с большим объемом данных	Программирование запросов	Планирование и проведение небольших исследований с помощью банка данных
12. Коммуникационные технологии*		Коммуникации посредством ЭВМ		Новая технология общения людей, проводящих исследование

* Программно-методическое обеспечение темы разрабатывается.

ки не могут дать определение информационных процессов, но они приводят множество содержательных примеров, замечают эти процессы в окружающем мире.

Очень важна форма проведения занятий. Это не монолог учителя, не традиционные объяснения и опросы, а беседы, обсуждение новых понятий, совместный поиск и анализ примеров, иногда переходящий в игру, иногда превращающийся в дискуссию.

Впрочем, беседами дело не ограничивается. Для закрепления теоретических понятий разработана серия упражнений. Рассмотрим некоторые из них.

Задача 1.

(Умение различать информационные процессы. Пропедевтика умения выделять существенные признаки явления.)

Котенок Васька завтракает в лесу

Пошел котенок Васька в лес позавтракать. Попробовал бруснику — кислая. Попробовал гриб-боровик — горький.

Видит — лежит в траве комочек взъерошенный. «Ну,— думает котенок,— тобой и позавтракаю».

Хотел он лапой подкатить комочек поближе, да как закричит: «Ой! Ай!» Комочек ежом оказался.

Сел Васька на пень и задумался. Думал он, думал и решил, что лучше с ежом не связываться, а поесть кислых ягод да горьких грибов.

Задание.

Где в рассказе происходит обработка, хранение и передача информации?

Задача 2.

(Умение обрабатывать информацию. Пропедевтика работы с базами данных.)

Пользуясь таблицей, ответьте на вопросы (на уроке таблица заполняется реальными сведениями об учениках класса):

1. Сколько человек живет на четвертом этаже?
2. Кто живет на втором этаже?
3. Кто живет на одном этаже с тобой?
4. Кто живет выше Маркиной?
5. Кто живет выше всех?
6. Сколько человек живет ниже пятого этажа?
7. Есть ли живущие на одном этаже с Житниковой?
8. Есть ли живущие на третьем этаже?
9. Сколько человек родилось в феврале?
10. Сколько человек родилось летом?
11. Кто родился в октябре?

Фамилия	Дата рождения	Этаж, на котором живет ученик
Большаков	23.04.76	2
Вершинин	17.02.76	7
Дратинский	30.11.75	1
Егорычева	17.02.76	2
Житникова	08.02.76	1
Зайченко	28.03.76	4
Колесникова	19.05.76	2
Коломиец	09.08.76	2
Кручинина	13.01.76	2
Маркина	07.08.76	6
Мельников	27.02.76	4
Мищенко	04.10.75	5
Нефедов	18.07.76	7
Новиков	11.06.76	5
Петрова	21.01.76	4
Рыкова	31.07.76	4
Фарфоров	08.10.76	8
Шаталина	03.08.76	5
Шопарова	10.10.75	4

12. Кто родился весной?
13. Сколько человек родилось раньше 10 февраля 1976 года?
14. Кто родился после 1 августа 1876 года?
15. Кто старше Дратинского?
16. Сколько человек младше Вершинина?
17. Есть ли родившиеся в сентябре?
18. Кто моложе тебя?
19. Кто самый младший?
20. Кто старше Маркиной и младше Рыковой?
21. Кто старше Шопаровой и младше Шаталиной?
22. Кто живет выше Большакова и старше его?
23. Кто младше, чем Коломиец, или живет ниже его?
24. Кто живет в самом высоком доме?

Это задание преследует множество учебных целей. Поиск ответов на вопросы заставляет школьника почувствовать обработку информации не как разовое действие, а как процесс. Возникает необходимость как-то упорядочить поиск, у школьника появляется потребность в алгоритмах, гарантирующих получение правильного ответа, и он самостоятельно открывает многие алгоритмы поиска информации.

Эта задача подводит и к разговору о понятиях полноты и избыточности информации. Например, для ответов на вопросы 1—21 нужны не все графы таблицы, а на вопрос 24 ответ дать нельзя.

Задача 3.

(Умение кодировать и декодировать информацию. Пропедевтика понятия необходимых и достаточных признаков предмета.)

Передай товарищу информацию в виде текстового сообщения, по которому он смог бы нарисовать точно такие же рисунки, как у тебя:

а) нарисован квадрат;

б) нарисован прямоугольник;

в) нарисованы четыре вертикальные палочки на равном расстоянии друг от друга:

Предполагается, что необходимые измерения проводятся школьниками самостоятельно с использованием метрической линейки. Учитель не говорит про измерения, но подсказывает, что линейка может потребоваться.

Задача 4.

Расшифруйте следующие сообщения и расскажите, как они были зашифрованы.

а) Ослышели уни шом, вбажели в думик и обили вулке.

б) Тутаску, почема а вуш букие тольсие заты?

Освоение клавиатуры ЭВМ. Знакомство с компьютером традиционно начинается с освоения клавиатуры. В Роботландии на это не отводится специальных занятий. Уроки-беседы завершаются 10—15 мин работы с компьютером. Это позволяет разнообразить формы работы на уроке, поддерживает интерес детей, обеспечивает соблюдение медицинских норм.

Традиционные клавиатурные тренажеры нацелены на запоминание размещения знаков алфавитной клавиатуры, однако это требует длительного времени тренировки и дает заметный эффект только при наборе достаточно больших текстов.

В то же время опыт показывает, что наибольшие трудности у начинающих вызывает не поиск нужной буквы, а установка режимов (переключение регистров и алфавитов, ввод цифр и знаков) и исправление допущенных ошибок.

В соответствии с этим освоение клавиатуры ведется двумя путями. Первый — работа с различными режимами. Режимы осваиваются при решении на компьютере алгоритмических задач. Программы построены так, что решение каждой задачи требует использования новых режимов работы клавиатуры.

Второй путь освоения клавиатуры — исправление ошибок. Для этого разработан специальный тренажер — Правилка. Работая с Правилкой, дети учатся определять вид ошибки (неверный символ, пропущенный символ, лишний символ) и исправлять ее.

Устройство ЭВМ — тема, которая логически завершает беседы об информации. Перед школьниками раскрывается сущность компьютера как универсальной информационной машины, они рассматривают составные части компьютера с точки зрения уже знакомых им процессов передачи, хранения и обработки информации.

Устройство компьютера рассматривается при этом чисто функционально: важно, что делает каждая составная часть машины, а не как она это делает.

На уроках рассматривается упрощенная схема, показанная на рис. 1. На схеме показаны основные устройства ЭВМ и их функциональное назначение. Для первого знакомства выбран минимальный набор устройств, только те, которые есть в ученических компьютерах. Определить функции дисковода, принтера, других устройств, добавить их к схеме можно при выполнении упражнений.

Красной нитью проходит через схему информация. Именно по отношению к ней рассматриваются функции всех устройств. Фактически это простейшая информационная схема компьютера. Информация вводится в машину с помощью клавиатуры и сохраняется в памяти. Из памяти информация поступает в процессор на обработку. Полученная в результате обработки новая информация снова сохраняется в памяти. Хранящаяся в памяти информация может быть выведена на монитор.

На основе этой схемы создан специальный тренажер. Схема изображается на экране компьютера. Глядя на нее, ученик рассказывает (учителю, товарищу, самому себе) о происходящих в компьютере информационных процессах. Затем надписи, соответствующие функциям устройств, гаснут. Необходимо повторить рассказ, восстанав-

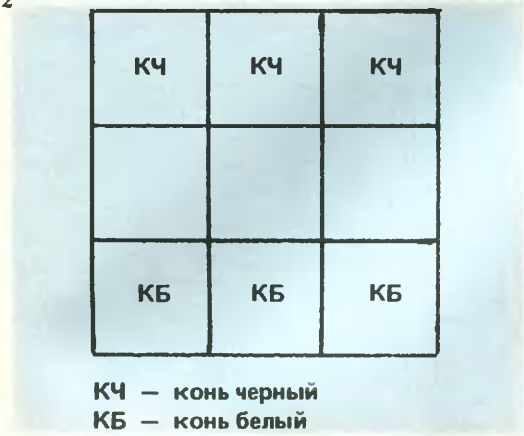
ливая недостающие слова на экране. Если все функции указаны правильно, гаснут названия устройств, их тоже необходимо восстановить.

Алгоритмические задачи. Задачи эти давно и широко известны. Вот несколько примеров алгоритмических задач, использованных в Роботландии:

1. Перевозчик. Человеку надо перевезти через реку волка, козу и капусту. В лодку, кроме человека, может войти либо только волк, либо только коза, либо только капуста. Нельзя оставлять на берегу без присмотра волка с козой или козу с капустой. Как человеку перевезти всех через реку?

2. Кони. Поменяйте местами белых и черных коней (рис. 2). Они ходят по шахматным правилам.

2



КЧ — конь черный
КБ — конь белый

3. Переливашка. Восьмилитровый сосуд наполнен водой. Пользуясь сосудами объемом 5 л и 3 л, разделите воду на 2 равные части.

Основная цель этих задач — формирование представлений об алгоритмах. На уроках математики школьник часто пользуется готовыми алгоритмами, но редко создает их. Он привык к тому, что задача должна иметь числовой ответ, и многие третьеклассники, правильно построив план перевозок через реку (задача 1), считали задачу нерешенной, так как не видели никакого окончательного ответа. Решение серии алгоритмических задач призвано сформировать представление о том, что ответом задачи может быть описание процесса, т. е. алгоритм.

Алгоритмические задачи задаются школьникам на дом, они решают их с помощью моделей, инсценируют в классе.

Некоторые задачи реализованы на компьютере. Основная цель здесь — дать возможность каждому ученику продемонстрировать решение задачи, а учителю — убедиться, что каждый ученик решил ее.

Но у этих простых программ есть и более дальние цели. Здесь ясно виден один из основных методических принципов курса — принцип опережения.

Во-первых, все программы решения алгоритмических задач реализованы как роботы-исполнители. У них есть среда и система команд, они различают синтаксические и семантические ошибки.

Во-вторых, работая с программой, дети фактически вынуждены записывать алгоритм в формализованном виде.

В-третьих, эти задачи иллюстрируют применение компьютера как инструмента моделирования ситуаций.

Таким образом, задолго до изучения соответствующих понятий у школьника складывается багаж примеров и формируются интуитивные представления о них.

Наконец, как уже говорилось, работая с этими программами, дети изучают клавиатуру компьютера.

Черные ящики появляются на уроке как игра в обработку информации. Ведущий задумывает алгоритм, остальные должны угадать его. Для этого они предлагают ведущему информацию, он обрабатывает ее по задуманному алгоритму и сообщает результат.

Вот примерный протокол игры:

слон	1
лягушка	0
корова	2
молоко	3

Алгоритм угадан: он считает буквы «о». (Примеры алгоритмов можно найти в [3].)

Черные ящики связывают линии информации и программирования. Поначалу игра рассматривается с информационной точки зрения. Это дает возможность поговорить о различных формах представления информации (слова, числа, рисунки...), о преобразовании этих форм при обработке информации.

По мере освоения игры (сначала учитель демонстрирует готовый протокол, затем дети угадывают задуманные им алгоритмы, наконец, они начинают сами придумывать их) школьники начинают понимать, что такое алгоритм обработки информации, они могут отличить хорошо заданный алгоритм от плохо заданного, исполняют довольно сложные алгоритмы.

Таким образом, черные ящики ведут учеников от линии информации к линии программирования.

С другой стороны, черные ящики — одно из самых значимых мест в линии творчества. Отгадывание черного ящика напоминает процесс создания научной теории. Сначала — наблюдения, опыты, накопление фактов. Затем выдвигается гипотеза. При этом приемлема только та гипотеза, которая

объясняет все имеющиеся факты. Гипотезу надо проверить. Для этого проводится еще несколько опытов, результаты которых необходимо предсказать. Если гипотеза позволяет правильно предсказать результаты, то не исключено, что она верна. Правда, в науке, в отличие от игры, никогда нельзя быть уверенным, что найден окончательный ответ.

Одновременно с черными ящиками рассматривается одна из важнейших тем курса — исполнители.

Идея использования исполнителей для обучения детей программированию восходит к работам С. Пейперта [8] и Г. Звенигородского [9]. Понятие исполнителя используется и в курсе информатики для старшеклассников [10, 11], и в подготовке профессиональных программистов [12].

В Роботландии исполнители появляются после решения алгоритмических задач. Решение этих задач на компьютере подготовило детей к пониманию формального исполнения.

На уроках обсуждаются общие свойства исполнителей: среда, система команд исполнителя (СКИ), виды ошибок. Исполнители фиксируют 2 вида ошибок: «Не понимаю» (синтаксическая ошибка) возникает при попытке дать исполнителю неверную команду, «Не могу» (семантическая ошибка), когда известная исполнителю команда не может быть исполнена в конкретной ситуации.

Эти понятия закрепляются игрой в исполнителя. Один из детей становится исполнителем, другие все вместе придумывают для него систему команд. Типичная система команд выглядит так:

РУКА — поднять руку;
СЕСТЬ — сесть на стул;
ПРЫЖОК — подпрыгнуть;
СМИРНО — встать по стойке смирно;
КОШКА — два раза мяукнуть.

Затем все по очереди отдают исполнителю команды. Он должен либо исполнить команду, либо ответить «Не могу» или «Не понимаю». Исполнитель должен быть очень внимателен, товарищи часто стараются «подловить» его: например, при вышеприведенной СКИ на команды СЯДЬ, ПРЫГНИ, МЯУКНИ надо ответить «Не понимаю».

Теоретические беседы и игры сочетаются с изучением новых компьютерных исполнителей. Это Машинист (идея исполнителя заимствована у Г. Звенигородского [9], но среда Роботландского Машиниста очень проста), Плюсик (простейший стековый калькулятор), Автомат (числовое устройство, умеющее умножать и складывать). При этом все исполнители рассматриваются по единой схеме (рис. 3).

Завершает эту тему информационный анализ исполнителей. Выясняется, что все исполнители описываются по единой информационной схеме (рис. 4). Эта схема сначала строится для очевидных частных случаев (Автомат, Плюсик), а затем обобщается на всех исполнителей. При этом происходит и обобщение понятия информации: информация здесь трактуется как описание состояния среды исполнителя.

Таким образом, исполнители идут как бы навстречу черным ящикам: от программирования к информации.

Работа с различными редакторами — одно из наиболее массовых применений современных персональных ЭВМ. Ведущее место занимают, конечно, редакторы текстов, растет применение графических, музыкальных, специализированных редакторов. Кроме того, редакторы часто включаются в различные интегрированные системы.

Ясно, что линия «компьютер» была бы неполной без освоения компьютерного редактирования. В Роботландии есть текстовый, графический и музыкальный редакторы.

Текстовый редактор Микрон разработан специально для школьников. Он не позволяет обрабатывать очень большие тексты, в нем нет многих возможностей развитых профессиональных редакторов. Однако основные функции редактирования (вставка, удаление, форматирование, копирование фрагментов текста) позволяют школьнику получить достаточно полное представление об использовании ЭВМ в работе с текстами.

Дополнительная возможность Микрона — система поддержки компьютерных диктантов. Компьютер сравнивает набранный учеником текст с образцом, который заранее подготовил учитель, и сообщает о сделанных школьником ошибках.

Графический редактор Раскрашка — это, по существу, редактор-конструктор. По экрану перемещается шаблон, который может изменять форму (форм всего 4: квадрат, круг, овал, треугольник), размеры и цвет. Оставляя в нужных местах след шаблона, можно получить рисунок. Раскрашка — очень простой графический редактор, средств которого вполне достаточно для формирования представлений о графических возможностях компьютера и реализации творческой фантазии младших школьников. В то же время построение нужной фигуры из ограниченного набора имеющихся шаблонов представляет задачу конструирования, иногда довольно непростую.

Шарманщик — музыкальный редактор Роботландии. В режиме, напоминающем редактирование текста, на нотном стане набираются ноты, образующие одноголосую мелодию. В процессе набора можно прослушать звучание каждой ноты, подобрать необходимую высоту и длительность, а когда мелодия будет набрана — услышать ее целиком.

Мы не ограничиваемся практическим освоением конкретных редакторов. Эта тема дает богатый материал для теоретических размышлений школьников.

Во-первых, редакторы дают повод поговорить об общих процессах создания и редактирования информации. Дети понимают, что они и раньше встречались с редакторами, приводят примеры: расстановка кораблей в игре «Морской бой», создание лабиринтов в игре «Мудрый кролик» [2], редактирование командной строки при работе с исполнителями.

Во-вторых, редакторы показывают многообразие форм информации. Рассматривая сходства и различия различных редакторов, дети приходят к пониманию общих свойств информации и отличительных особенностей ее конкретных форм.

В-третьих, редакторы — это исполнители. У них есть система команд, есть среда, есть реакция на синтаксические и семантические ошибки. На этом примере школьники видят новые для них способы управления исполнителями, учатся замечать общую сущность за различными внешними проявлениями.

Анализ редакторов как исполнителей имеет еще одну положительную сторону. Начинающие пользователи ЭВМ часто не видят различия между действием и клавишей, при нажатии на которую оно выполняется. У них формируется «кнопочное мышление», имеющее ряд негативных последствий. Выполняемые преобразования планируются в терминах кнопок, а не действий, что делает практически невозможным сознательное выполнение сколько-нибудь длинной цепочки действий. При освоении нового программного продукта человек, вместо того, чтобы разобраться в его принципиальных возможностях, стремится запомнить все управляющие клавиши.

При рассмотрении редактора как исполнителя внимание акцентируется на командах, а не на клавишах. Школьники понимают, что нажатие на клавишу — всего лишь способ отдать команду исполнителю. Приемы редактирования рассматриваются не как последовательности нажатий, а как алгоритмы, что способствует быстрому формированию и сознательному использованию этих приемов.

(Интересный подход к проблеме «кнопочного мышления» предложен в системе LOGOWRITER [13] — одной из последних реализаций языка Лого. У каждой «кнопочной» команды в этой системе есть двойник — «текстовая команда». Таким образом, редактирование можно вести в командном режиме или даже запрограммировать.)

Для иллюстрации подхода к изучению редакторов приведем фрагмент реального урока, который состоялся после того, как дети освоили практическую работу с Раскрашкой.

УЧИТЕЛЬ. На предыдущих уроках мы познакомились с Раскрашкой. Вы знаете, что с помощью Раскрашки можно рисовать различные картинки и даже мультфильмы. А как вы думаете, можно ли считать Раскрашку исполнителем?

СЕРЕЖА. Я думаю, что Раскрашка — не исполнитель.

УЧИТЕЛЬ. Почему?

СЕРЕЖА. Ну вот, например, когда мы работали с Плюсиком, мы набирали команды, а у Раскрашки команд нет.

МИША. Раскрашка не исполнитель, потому что у нее нет системы команд.

ТАНЯ. А мне кажется, что у Раскрашки все-таки есть команды.

ОЛЯ. Раскрашка, наверное, не исполнитель, потому что она не создает новой информации. Вот Плюсик — он что-то считает, а Раскрашка только показывает рисунок, который мы сами придумали.

ЛЮСЯ. Рисунок — это тоже информация.

ЛЕНА. Мне кажется, что Раскрашка — это исполнитель, потому что у нее есть система команд.

УЧИТЕЛЬ. Что-то мы немного запу-

тались. Так, есть у Раскрашки команды или нет?

МИША. Есть. Только они задаются не словами, а кнопками.

УЧИТЕЛЬ. Давайте назовем эти команды!

ЮРА. Стрелки.

УЧИТЕЛЬ. Стрелка — это команда?

ЮРА. Нет. Стрелки — это клавиши. Если их нажать, то будет команда — «подвинуть курсор».

УЧИТЕЛЬ. Хорошо. Только мы договорились называть курсор у Раскрашки шаблоном. Еще какие команды вы знаете?

(Дети называют команды: «Изменить цвет», «Закрасить», «Повернуть шаблон», «Изменить шаблон», «Уменьшить», «Увеличить».)

УЧИТЕЛЬ. Так можно ли считать Раскрашку исполнителем?

ТАНЯ. Раз есть система команд — значит, можно.

УЧИТЕЛЬ. Но ведь мы не набираем никаких слов!

ЛЮСЯ. Команды не обязательно отдавать словами. Например, светофор дает команды водителям своими огнями, а милиционер — жестами. А Раскрашка понимает команды-кнопки.

УЧИТЕЛЬ. Обратите внимание, это очень важный вывод. Команды можно задавать не только словами, но и кнопками, жестами, звуками. Важно только, что у исполнителя есть система команд, он умеет различать (неважно, в какой форме) и выполнять их. А как вы думаете, бывают ли у Раскрашки ситуации «не понял» и «не могу»?

МИША. Наверно, нет, потому что Раскрашка не дает таких сообщений.

СВЕТА. А их тоже не обязательно давать словами. Вот Кролик ничего не говорил, а просто упирался в стенку.

НАТАША. У Раскрашки бывает «не понял»; когда, например, нажмешь клавишу, где нет никакой команды, Раскрашка ничего делать не будет...

УЧИТЕЛЬ. А как насчет «не могу»?

ЛЕНА. Если мы будем увеличивать или уменьшать шаблон, то это можно делать не бесконечно, а только до какого-то размера. А потом нажимаешь — и ничего не получается. Это «не могу».

УЧИТЕЛЬ. Итак, мы установили, что Раскрашка, конечно же, исполнитель, раз у нее есть даже «не понял» и «не могу». А работает ли этот исполнитель с информацией?

ДИМА. Работает, только с картинками.

УЧИТЕЛЬ. А картинки — это информация?

ДИМА. Я думаю, что да.

УЧИТЕЛЬ. Мы с вами встретились с новой формой представления информации — графической. А какие еще формы вы знаете?

АНДРЕЙ. Цифры, буквы...

УЧИТЕЛЬ. Принято говорить в этих случаях — «числовая и текстовая информация». А какие из наших исполнителей работали с этими формами представления информации?

АНДРЕЙ. Плюсик — с числовой. Микрон — с текстовой.

УЧИТЕЛЬ. Совершенно верно. А теперь включите компьютеры!

Практической проверке программных и методических средств Роботландии посвящена серия педагогических экспериментов.

Впервые некоторые ключевые исполнители и методические приемы были опробованы на Переславской летней школе юных программистов в августе 1987 г. [2]. С сентября 1987 г. в нескольких классах двух переславских школ начат сравнительный педагогический эксперимент. Одна из его целей — оценить возраст учеников, начинающих изучать информатику. Двухлетний курс Роботландии начат одновременно в III и в V классах. Эксперимент еще не завершен, но уже накоплен достаточный материал для обсуждения конструктивных рекомендаций.

Летом 1988 г. в рамках Международной детской компьютерной школы под Переславлем работала кафедра Роботландии, где эта программно-методическая система испытывалась в условиях внешкольных форм учебно-воспитательной работы.

В 1988/89 учебном г. в Переславле-Залесском и нескольких других городах страны начат расширенный эксперимент по преподаванию двухлетнего общеобразовательного курса информатики на базе Роботландии. В Переславле этот курс поставлен в третьих классах двух городских школ. Уроки информатики, включенные в расписание третьих классов, проводятся дважды в неделю, каждый класс разбит на две подгруппы. Ведут занятия разработчики Роботландии и школьные учителя. Работа ведется с пробным учебником и пособием для учителя с поурочными методическими комментариями. Оперативный контроль эксперимента, анализ уроков, обсуждение ближайшей перспективы осуществляет еженедельный семинар учителей-экспериментаторов.

Литература

1. *Дуванов А.* Информатика без ЭВМ // Информатика и образование. 1987. № 3.

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

3. *Дуванов А., Гольцман М.* Черные ящики / Информатика и образование. 1988. № 5.

4. *Ершов А. П., Звенигородский Г. А., Пер-*

шин Ю. А. Школьная информатика (концепции, состояние, перспективы): Препринт № 152. Новосибирск, 1979.

5. *Амбрасас Ю.* Письмо в редакцию / Информатика и образование. 1988. № 3.

6. *Звенигородский Г. А.* Система математического обеспечения, ориентированного на школьный учебный процесс // УСиМ. 1980. № 5.

7. *Бърнев П.* Информатиката в средното училище // Обучението на математиката. 1979. № 5.

8. *Papert S.* Mindstorms, Children, computer and powerful ideas. The Harvester Press Ltd., 1981.

9. *Звенигородский Г. А.* Первые уроки программирования. М., 1985 (Б-чка «Квант». Вып. 41).

10. *Ершов А. П., Кушниренко А. Г., Лебедев Г. В.* и др. Основы информатики и вычислительной техники. М.: Просвещение, 1988.

11. Основы информатики: Методическая разработка. Свердловск, 1987.

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

13. LogoWriter // Reference Guide. Logo Computer Systems Inc., 1986.

Заказы на поставку Роботландии (программное обеспечение, учебник, пособие для учителя и другие материалы) можно направлять по адресу: 152140, г. Переславль-Залесский, а/я 46, Роботландия.

В настоящее время программное обеспечение разработано для «Ямахи», возможен перенос на другие машины («Корвет», УКНЦ, IBM-PC) по заказу.

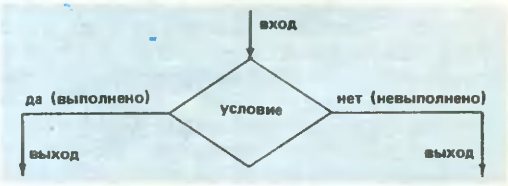
П. АСЕНОВА

Изучение алгоритмической конструкции выбора вариантов

Алгоритмизация — это один из основных разделов курса информатики, в котором выделяются следующие содержательные линии: понятие об алгоритме, основные алгоритмические конструкции, величины и этапы решения задач. Ключевую позицию среди алгоритмических конструкций занимает выбор вариантов, который трудно усваивается учащимися. Данная статья представляет методику изучения этой алгоритмической конструкции как часть методики обучения алгоритмизации в рамках двух учебных часов.

Содержание темы нацелено на ознакомление учащихся с алгоритмической конструкцией выбора и формирование умений использовать ее для принятия решений. Очерчиваются следующие информационные ядра: конструкция выбора — общий вид и исполнение; логические выражения в качестве условий — операции отношений, простые условия, логические операции, сложные условия; дерево решений. Учебная деятельность способствует формированию следующих умений: понимать целесообразность использования конструкции выбора; исполнять, изменять и составлять алгоритмы для различных исполнителей, содержащие конструкции выбора; составлять логические выражения в качестве условий и определять из значение; устанавливать эквивалентность или неэквивалентность алгоритмов, содержащих конструкции выбора; искать решение по дереву решений и строить его; проверять полноту решений по дереву; проверять правильность работы алгоритма путем чтения или исполнения для подобранных наборов данных; применять конструкцию выбора для решения различных задач.

Мотивационная сторона введения конструкции выбора раскрывается на фоне неизбежности принятия решений человеком. Информатика и вычислительная техника помогают ему с большей скоростью сравнивать множество вариантов и выбирать оптимальный из них. Обычно принятие того или иного решения зависит от некоторого условия. В программировании оно обозначается следующим блоком (рис. 1):



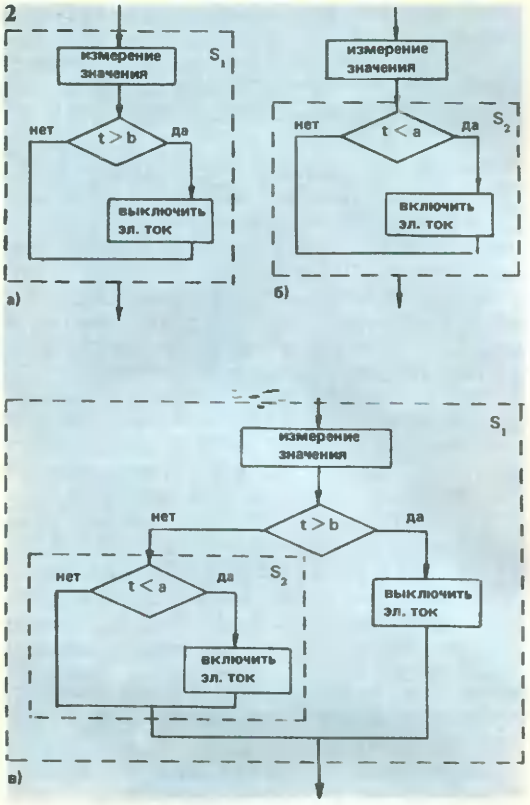
1

Простейшие случаи принятия решения вводятся на следующем примере:

Задача 1. Терморегулятор контролирует изменение температуры какого-то процесса. Опишите его действие, при котором он следит за тем, чтобы температура: а) не была выше b °С; б) не была ниже a °С; в) оставалась в интервале $[a, b]$ °С.

Решение приводится на рис. 2:

На рис. 2а и 2б видно, что регулятор выбирает один из двух вариантов — выполнить данное действие или пропустить его — в зависимости от некоторого условия.



Конструкция выбора такого типа представлена на рис. 3а. Она записывается следующим образом:

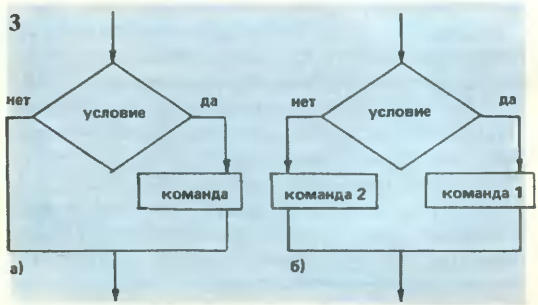
ЕСЛИ условие ТО команда

и выполняется так: анализируется условие и, если оно выполнено (принимает значение «да»), команда выполняется, в противном случае она пропускается.

На рис. 2в терморегулятор выбирает одно из двух действий в зависимости от условия. Конструкция выбора такого типа представлена на рис. 3б. Она записывается следующим образом:

ЕСЛИ команда ТО команда 1 ИНАЧЕ команда 2

и выполняется так: анализируется условие и, если оно выполнено (принимает значение «да»), выполняется командой 1, в противном случае — команда 2.



Команды могут быть составными (см. рис. 2в):

От учащихся целесообразно потребовать привести примеры принятия решений. В процессе обсуждения подчеркивается их роль при регуляции биологических, технических, социальных и прочих процессов. В качестве иллюстрации приведем несколько примеров.

Электронные схемы монтируются в разных приборах (хозяйственных, медицинских, транспортных, производственных и т. д.). Они регистрируют контролируемые параметры (например, скорость, давление, температуру, концентрацию) и передают их специальным устройствам, которые в зависимости от значения параметров вырабатывают то или другое воздействие. Таков принцип действия автопилота, искусственных почек, аппарата искусственного кровообращения и т. д.

В соответствии с современными физиологическими представлениями терморецепторы в кожных покровах регистрируют изменения температуры окружающей среды. Нервным путем сигналы передаются гипоталамусу, вследствие чего поддерживается стабильная температура тела. Если жарко, начинается отделение пота, приток крови к коже усиливается и активность мышц понижается. Если холодно, по коже пробегает дрожь и мышечная активность повышается.

Далее идет учебная работа по закреплению конструкции выбора.

Задача 2. Рассмотрите алгоритм:

1. Ввод значений a и x;
2. $y1 := (x^2 + a^2) \cdot (x - a)$;
3. $y2 := a \cdot x / (x - a)$;
4. $y := y1 + y2$;
5. Вывод значения y;
6. Конец.

Какую задачу решает алгоритм? Исполните его для: $a=4$, $x=2$; $a=3$, $x=3$. Какова область применения этого алгоритма? Какие изменения обеспечивают его работу для любых значений a и x?

Решение этой задачи показывает роль конструкции выбора для проверки корректности данных.

Задача 3. Опишите алгоритм для

а) проверки равенства двух чисел (слов);

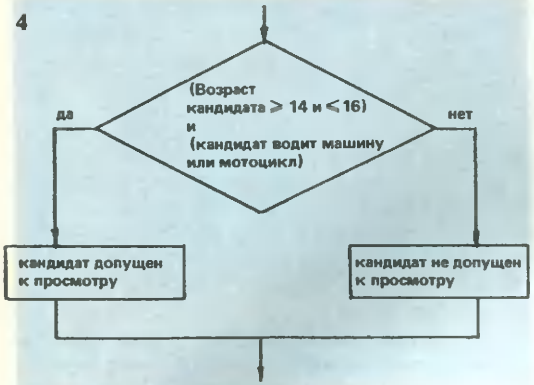
б) вычисления абсолютного значения числа x.

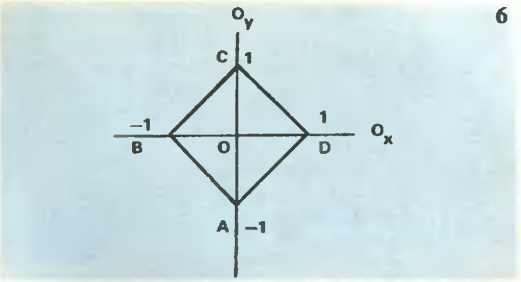
Задача 4. Для съемки фильма требуется подросток 14—16 лет, умеющий водить машину или мотоцикл. Опишите схему, по которой допускают или не допускают кандидата к отбору.

Учащиеся могут решить эту задачу, используя приобретенные знания и простые ус-

ловия. Последние образуются чаще всего с помощью операций отношений ($>$, $\geqslant$, $<$, $\leqslant$, $=$, $\neq$). Однако такое решение слишком сложное. Его можно упростить, если все требуемые условия проверяются как бы одновременно (рис. 4). На основе этой задачи вводятся операции отношений и логические операции И, ИЛИ, НЕТ. Потом они закрепляются в задачах.

4





6

а) т. P совпадает с координатным началом;

б) т. P принадлежит координатным осям;

в) т. P не принадлежит координатным осям;

г) т. P принадлежит первому или третьему квадранту;

д) т. P принадлежит прямой CD;

е) т. P принадлежит отрезку CD;

ж) т. P принадлежит первому квадранту, но не принадлежит отрезку C;

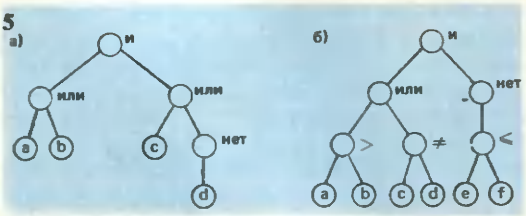
з) т. P принадлежит границе квадрата.

Задача 8. Являются ли эквивалентными следующие алгоритмические фрагменты:

А	если А то Б иначе В	если не А то В иначе Б
Б	если А<Б то Н=Х иначе Н=У	если А<Б то Н=Х Н=У
В	если Х<А то У=Т У=Н	У=Н если Х<А то У=Т
Г	если Х<А то У=Т иначе У=Н	У=Н если Х<А то У=Т
Д	если Х<А и У=Н то Р=Т	У=Н если Х<А то Р=Т
Е	если У=Т или У=Н то Р=М	У=Т Р=М

Задача 5. Какие логические выражения представлены на рис. 5?

5



Задача 6. Определите значения (да/нет) следующих логических выражений:

а) А⩽В или А=В−1, где А=0,5 и В=3/7;

б) А>В или А⩽В, где А и В — численные переменные;

в) «А+В»=«А»+«В»;

г) «А»+(«В»+«С»)=(«А»+«В»)+«С» и «А»+«В»=«В»+«А»;

д) нет (А/В≠1) и А=В, где А и В — численные переменные.

Задача 7. Если т. P на плоскости имеет координаты X и Y, запишите следующие логические выражения (см. рис. 6):

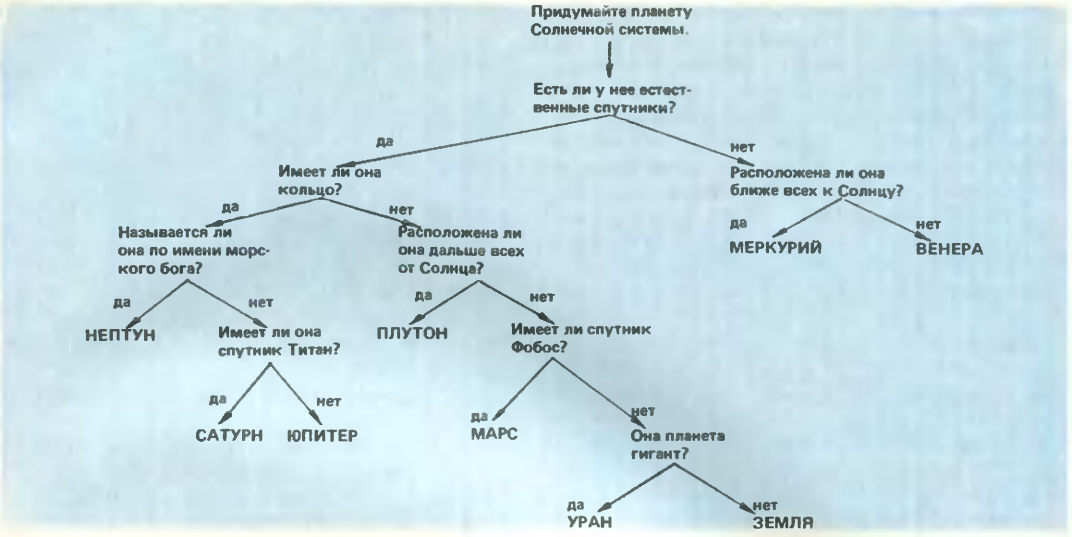
Рассмотренные разновидности конструкции допускают выбор между двумя возможностями. Их число увеличивается при последовательном соединении конструкции выбора. Таким образом получается дерево возможных решений для построения алгоритма (дерево алгоритма). Конкретные решения находятся на самом нижнем уровне дерева.

Задача 9. Рассмотрите дерево алгоритма угадывания задуманной вами планеты Солнечной системы (см. рис. 7). Придумайте планету и проследите за исполнением алгоритма.

В процессе решения обращается внимание, что дерево решений вносит наглядность в сложные ситуации. Оно позволяет не только легко выбрать решение, но и проверить полноту решений и корректность по содержанию. В рассмотренном примере на дереве видна полнота — представлены все планеты Солнечной системы; содержательную корректность можно проверить, используя справочники, энциклопедии и т. п.

Задача 10. Постройте дерево алгоритма следующей задачи. Измеряется и вводится

7



48 температура человеческого тела. В зависимости от данных устройство выдает соответствующее сообщение. Опишите алгоритм решения задачи. Определите набор данных, которые проверяют работу алгоритма по всем ветвям его дерева.

Примечание: необходимые данные учащиеся найдут в медицинской энциклопедии, БСЭ или других справочниках.

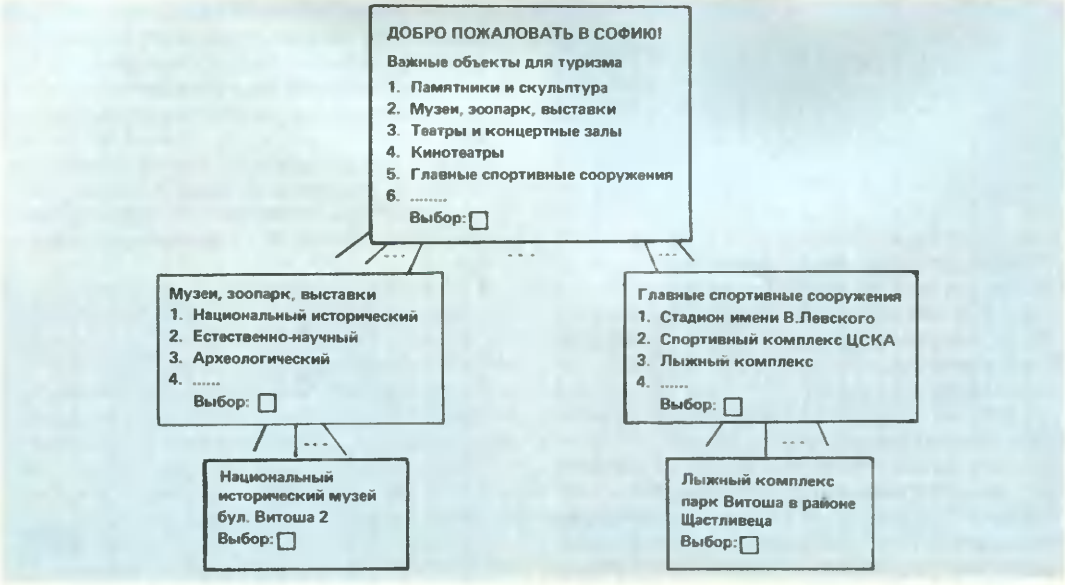
Задача 11. Для гостей Софии участники кружка информатики сделали компьютерный справочник основных туристических объектов города. Структура программы показана на рис. 8. Рассмотрите ее и ответьте на следующие вопросы: а) какую информацию можно получить из этой программы; б) какова структура программы; в) каким образом осуществляется переход к каждому последующему уровню; г) что надо сделать, чтобы получить информацию о лыжном комплексе?

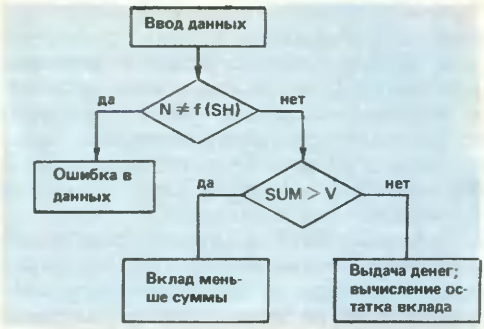
Сделайте небольшой компьютерный справочник в выбранной вами области.

Задача раскрывает роль алгоритмической конструкции выбора для реализации систем по принципу меню.

Задача 12. Опишите алгоритм для банковского автомата. В щель автомата клиент вставляет свой личный бланк, на котором находится шифр (число) SH, набирает номер счета N и размер требуемой суммы SUM для получения. Данные вводятся в центральную ЭВМ, которая проверяет соответствие

8





шифра номеру счета предъявителя, проверяет наличный вклад V в банке и, если вклад позволяет, подает сигнал банковскому устройству для выдачи требуемой суммы.

Примечание: $\bar{f}(SH)$ — формула соответствия шифра SH номеру N.

Статья не привязана к конкретному языку программирования, чтобы учителя могли легче ее адаптировать к своим условиям работы.

В НРБ проведен эксперимент, который показал эффективность рассмотренной методики в обучении алгоритмизации.

Литература

1. *Асенова П.* Алгоритми — примери и задачи. София: МНП, 1988.

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

3. *Науман Э.* Принять решение — но как? М: Мир, 1987.

4. Человек и машина // Популярная энциклопедия: Пер. с англ. М.: Мир, 1986.

Н. АЛИПОВ, А. СОКОЛОВ
2-й медицинский институт, Москва

Организация контроля знаний

Экономия времени преподавателей, простота разработки методов наряду с высокой эффективностью, объективностью, продуманность и четкая постановка вопросов — таков неполный список преимуществ программированного контроля. С внедрением в педагогический процесс компьютеров программированный контроль получил новый, мощный импульс и сегодня широко используется во всем мире.

Однако его организация как в машинном, так и в безмашинном варианте сталкивается с определенными трудностями. В машинном варианте — это интерпретация свободно конструируемых ответов. Компьютер легко распознает альтернативные ответы (выбор одного из нескольких предлагаемых вариантов, ответы типа «да — нет» и т. д.), а также результаты математических вычислений, и поэтому именно такого рода вопросы широко используются в большинстве систем автоматизированного контроля. Разновидностью такого метода (хотя и весьма оригинальной, удобной и эффективной) является система, при которой в предъявляемом на экране дисплея тексте пропущено некое ключевое слово и обучающийся выбирает это слово из жестко установленного «словаря». Есть и другие разновидности тестов с альтернативными вопросами. Однако такого рода вопросы далеко не всегда могут исчерпывающе выявить уровень знаний студентов.

Для решения проблемы интерпретации свободно конструируемых ответов используется целый ряд методик: распознавание по ключевым словам или буквосочетаниям, использование элементов искусственного интеллекта, позволяющих интерпретировать смысловые структуры человеческой речи, в том числе вводимой не с пульта, а через микрофон.

Однако такие системы трудоемки, дорогостоящи, порой требуют достаточно мощных машин, при этом далеки от совершенства. Во-первых, нельзя предусмотреть все возможные варианты хотя бы чисто орфографических ошибок, и поэтому иногда машина может принять такую ошибку за неверный по существу ответ. Во-вторых, невозможно предугадать все существующие синонимы и стилистические обороты речи и поэтому даже самые совершенные системы иногда не распознают правильный, но оригинально построенный ответ. В-третьих, порой по содержанию ответ студента может быть совершенно неожиданным, но верным.

В безмашинном варианте этот основной недостаток устраняется путем самоконтроля студентов с последующей выборочной проверкой объективности их самооценки преподавателем.

Такая форма контроля позволяет студентам свободно строить ответы, а это очень существенное преимущество. Однако имеется

и целый ряд недостатков, свойственных безмашинным методикам: достаточно большие затраты времени, трудности со сдачей контроля в индивидуально выбранное время, а не одновременно со всей группой, и т. д. Кроме того, существенным недостатком, на наш взгляд, является то, что круг задаваемых вопросов известен студентам заранее. Это приводит к тому, что уже после «первого поколения» студентов раздаваемые им на руки методички с вопросами становятся исписанными ответами и в последующем подготовка к программированному контролю превращается из обдумывания вопросов в механическую зубрежку готовых ответов (не говоря уже о возможности прямого списывания со шпаргалки). В то же время студентам все-таки необходимо знать, о чем их будут спрашивать; в противном случае отпадает мощный стимул подготовки к занятиям, направляющий и ориентирующий студентов в поисках нужной информации в учебнике и специальной литературе.

Мы разработали вариант машинного контроля знаний, в котором пытались устранить (или свести к минимуму) многие описанные выше недостатки. Программы написаны на языке Бейсик для компьютерного класса КУВТ «Корвет». Для организации контроля использована программа локальной сети. Основные черты нашей системы программированного контроля:

вопросы могут быть как альтернативными, так и конструируемыми;

студенту заранее известно, *о чем* его спросит машина, но неизвестно, *как* она его спросит;

контроль может осуществляться индивидуально, а не одновременно со всей группой;

исключена неверная оценка ответов компьютером, обусловленная стилистическими, орфографическими и прочими ошибками, а также оригинальной (но верной) формулировкой ответа;

программы чрезвычайно просты, быстро составляются, могут быть легко мофицированы или дополнены и пригодны для массового применения.

В системе использованы следующие педагогические и программные решения:

1. Для подготовки к контролю студентам выдается не перечень вопросов, а те положения, в пределах которых они должны свободно ориентироваться. Если в «классическом» варианте программированного контроля студенту-медику для подготовки давался вопрос: «Подчиняется ли сердце закону «все или ничего»?» (закону, согласно которому сила сокращений сердца не зависит от силы его раздражения), то в нашем варианте, напротив, студенту в перечне необходимых знаний выдается положение: «Сердце подчиняется закону «все или ничего», а вопрос может быть сформулирован самым различным образом, например: «Может ли врач увеличивать силу сокращений сердца у больного, которому вживлен электрокардиостимулятор, изменяя амплитуду стимулирующих импульсов?»

2. Основной блок контролирующей программы состоит из нескольких (обычно около 10) списков вопросов. Каждый из этих списков соответствует одному из тех положений, которые входят в перечень, выдаваемый студенту. В каждом списке имеется множество (не менее 10) вопросов. В программе имеется подпрограмма «случайной выборки». Этой подпрограмме последовательно предъявляются все списки, и из каждого она выбирает один вопрос. Таким образом, студент обязательно опрашивается по всему перечню необходимых знаний, но «трасса» его прохождения по программе всегда разная.

3. В программе имеются вопросы самого разного типа — как альтернативные, так и конструируемые. В случае конструируемых вопросов в память вводится эталонный ответ и несколько наиболее вероятных его «синонимов». Если ответ студента не совпадает ни с одним из этих «синонимов», то студенту (после того, как он введет свой ответ) выдается эталонный ответ и надпись:

«Если Вы считаете, что ответили верно, введите «В».»

«Если Вы считаете, что ответили неточно, введите «Н».»

«Если Вы считаете, что ответили ошибочно, введите «О».»

За верный ответ ставится балл, за неточный — 0,5 балла, за ошибку — 0. Таким образом, студент сам себя оценивает. Весь этот диалог запоминается и после окончания контроля вместе с фамилией, номером группы студента, его оценкой и датой прохождения контроля либо выводится на дисплей, либо записывается на диск, либо распечатывается. Практика показала, что оценка преподавателем объективности самоконтроля студента занимает меньше минуты; студенты, узнав о стопроцентной проверке преподавателем диалога, перестают «жульничать» и завышать свои оценки. Данный прием заимствован из безмашинного программированного контроля, в течение многих лет успешно применяющегося на кафедре физиологии 2-го медицинского института; однако если в безмашинном варианте из-за временных ограничений преподаватель мог лишь выборочно проверять тетради отдельных студентов, то в компьютерном варианте проверка стопроцентна.

4. Студент может проходить контроль абсолютно самостоятельно, в любое удобное для него время. Для преподавателя достаточно просмотреть либо запись диалога и оценки на диске, либо распечатку.

Данный вариант контроля (на оценку «удовлетворительно») использовался нами как текущий контроль знаний для каждого занятия, и прохождение его было обязательным условием отработки данной темы.

Следует отметить, что эта методика в целом существенно повышает уровень требовательности к студентам. Так, для получения оценки «удовлетворительно» студенты должны дать такие ответы на вопросы, которые ранее оценивались «хорошо»; практика показала, что студенты «подтягиваются» и в целом успешно проходят контроль уровня «удовлетворительно». Однако очень интересно было бы проверить «выживаемость» знаний, так как это, по нашему мнению, важнейший показатель, характеризующий эффективность педагогических методик. Для этого мы выбрали две группы студентов, обучавшихся на одном потоке, слушавших одни и те же лекции; но одна группа проходила контроль по вышеописанной методике, другая — по обычной форме. Результаты обработаны статистически с использованием функции Фишера и критерия Стьюдента.

У опытной группы средний балл и процент прошедших контроль оказался (с достоверностью $P<0{,}01$) выше, чем у контрольной.

Таким образом, данная форма контроля, во-первых, приводит к существенной экономии времени. В среднем при сформировавшемся навыке работы с ЭВМ студент тратит не более 10 мин на ответы на 10 вопросов, а контроль преподавателя осуществляется за несколько секунд и поэтому может быть стопроцентным. Во-вторых, этот вариант контроля способствует существенному повышению «выживаемости» знаний. Кроме того, и это кажется нам немаловажным, интерес студентов к таким контролирующим программам чрезвычайно велик; порой их просто нельзя оторвать от компьютера, они «проходят» одну программу за другой, и это, по нашему мнению, неизмеримо лучше, чем ситуация, при которой взрослые люди прячут в партах шпаргалки и учебники и воспринимают любые виды контроля как насилие и неприятную необходимость.

Адрес для справок: 117437, Москва, ул. Островитянова, 1, 2-ой МОЛГМИ, кафедра нормальной физиологии, Алипов Николай Николаевич. Тел. 434-40-01.

Литература

1. *Александров Г. Н., Ратнер Г. Л.* Программированное обучение и его место в медицинских вузах. М.: Медицина, 1968.

2. *Крэм Д.* Программированное обучение и обучающие машины / Пер. с англ. М.: Прогресс, 1965.

3. Методические указания к составлению объективного тестового экзамена для оценки знаний студентов по клиническим дисциплинам / Под ред. И. А.Сыченикова. М.: Изд-во 1-го мед. ин-та им. И. М. Сеченова, 1982.

4. Программированный контроль по курсу физиологии для студентов лечебного факультета / Под ред. Г. И. Косицкого, А. В. Губарь. М.: Изд-во 2-го МОЛГМИ им. Н. И. Пирогова, 1978.

5. *Frigerio N. A., Stowe R. S.* Computer production of randomized examinations. Journal of Medical Education. 1974. V. 49. N 3.

6. *Harless W. G., Zier M. A.*, Duncan R. S. A voice-activated, interactive videodisc case study for use in the medical school classroom. Journal of Medical Education. 1986. V. 61. N 11.

7. *Muller J.* (chairman). Physicians for the 21 century / Report of the project panel on the general professional education of the physician and college preparation for medicine. Journal of Medical Education. 1984. V. 59. Part 2.

8. *Posteraro R. H., Blackwill E., Huddleston A. C.* TECHSCORE: a program for tabulating the results of multiple choice questions and correcting multiple choice examinations. Computers in Biology and Medicine. 1986. V. 16. N 4.

Бой выигран!

На всемирном турнире 1988 г. по компьютерной игре
«Бой в памяти»
(о ней было рассказано в «ИНФО» № 1 за 1988 г.)
победила программа советского участника
из Переславля-Залесского
Е. Лилитко.
Поздравляем!

КАБИНЕТ ВТ

А. АРХАНГЕЛЬСКИЙ

Мир ЭВМ

12. Форматирование текстов

Чем бы ни занимался человек, использующий ЭВМ, текст еще долго будет оставаться одним из основных видов отображения информации. Не нужно убеждать, что аккуратно отпечатанный текст с ровными правым и левым краями, одинаковыми отступами и разделенными страницами, с собранным, как в книге, содержанием и аннотацией воспринимается значительно легче и лучше, чем тот же текст, написанный от руки. Программа подготовки документов DOC позволяет достичь всего этого, более того, она позволяет в какой-то мере автоматизировать сборку в один документ разрозненных текстов, набранных ранее. Она позволяет однажды набранный текст использовать и отдельно, например в виде письма или статьи, и в то же время несколько отдельных текстов (в виде файлов) собрать в один с общим содержанием и общей нумерацией страниц. При этом не потребуются какие-либо дополнительные действия с исходными или выходными файлами. Это позволяет экономить время и распараллеливать работу как в пространстве, так и во времени.

Фирма DEC поставляла программу DOC (document) вплоть до четвертой версии. Затем, с разработкой новой системы подготовки текстов RUNOFF, такого рода системы стали называться текстовыми процессами (wordprocessor) и были выделены в неподдерживаемые утилиты.

Программа DOC поставлялась в таком виде, чтобы пользователь с помощью ответов на вопросы мог легко подстроить ее под свои требования. Поэтому появилось значительное количество версий; мы будем описывать ту версию, которая поставляется в ОС РАФОС и ОС ДВК. Коррекции, которые вносились в нее с 29 мая 1979 г. до 23 июня 1981 г., касались в основном титульных листов, листов утверждений и содержания в соответствии с советскими стандартами на оформление документации. Средства, которые читатель уже имеет в своем распоряжении, помогут ему найти отличие своей версии от нашей.

Определим те понятия, которыми мы будем оперировать. Мы впервые сталкиваемся с понятием метафайла. Метафайл — это файл, который перед использованием требует преобразования. В нашем случае это текстовый файл, содержащий некоторые директивы для программы DOC; прежде чем напечатать текст, файл необходимо преобразовать с помощью программы DOC. В остальном это такой же текстовый файл, как и другие; он создается любым текстовым редактором. Чтобы как-то отличить текст пользователя от указаний программе DOC, необходим специальный признак, указывающий на директиву, а также определение синтаксиса директивы. Вот оно:

директива может занимать только одну строку;

директива может иметь числовой или текстовый аргумент, отделенный от нее пробелом;

признаком директивы является точка в начале строки. Необходимо следить, чтобы в тексте пользователя такая комбинация символов не возникала. Чтобы избежать ее, достаточно поставить пробел перед точкой, оказавшейся в начале строки;

все директивы для программы DOC состоят из четырех символов, первый из которых — точка.

Окончание. Начало см.: Информатика и образование. 1987. № 1—3; 1988. № 3—6; 1989. № 4.

12.1. Директивы размещения текста

Указывать, что текст должен быть выровнен по левому и правому краям, нет нужды — это разумеется само собой. Но необходимо, например, указать, что начинается новый абзац, т. е. что далее текст идет с новой строки и с отступом (красная строка). Для этого служит директива LINE — строка:
.LIN N
Аргумент N указывает, сколько строк нужно пропустить перед абзацем (по умолчанию N=1). Он может изменяться от 0 до 57 (количество строк на странице). Если новый абзац начинается на другой странице, то пропуска строк от начала страницы не происходит.

Однако часто нужно, чтобы абзац начинался с начала строки (без отступа), например после рисунка. Тогда используется директива BEGIN LINE — начало строки:
.BLN N
В остальном она совпадает с директивой .LIN.

Ну а если нужно перепрыгнуть через несколько строк, например для того, чтобы потом вклеить фотографию? При этом в тексте должно оставаться место, даже если будет переход на следующую страницу. Подходящее английское слово для этой директивы — SKIP (прыжок):
.SKI N
Если произойдет переход на новую страницу, то DOC оставит свободными N строк начиная с начала страницы.

Ну а если вместо фотографии мы хотим создать рисунок, используя для этого обычные символы (как это можно сделать на обычной пишущей машинке)? Нужно средство, позволяющее на время отказаться от форматирования. В качестве директивы для этого используется сокращение слов SHOW TEXT.

Этот текст программой DOC не форматировался, потому что перед ним в начале строки стояла директива .SWT2

(двойка здесь обеспечивает пропуск двух строк перед текстом).

У директивы .SWT есть свои ограничения:

если текст шире, чем принятая ширина колонки, то он будет соответственно обрезан;

текст не должен содержать символы горизонтальной табуляции HT;

текст не должен содержать пустые строки (т. е. содержащие только символы ВК ПС); чтобы пропустить строку, она должна содержать хотя бы один пробел.

Действие директивы .SWT заканчивается, как только встретится какая-либо другая директива.

Если нужно напечатать большую таблицу, широкий листинг программы или что-нибудь подобное, нужно перейти на новую страницу, чтобы у нас в распоряжении был полный лист, и использовать директиву .LST (LIST). При этом снимаются ограничения на длину строки — она может состоять из 128 символов, а также может содержать символы горизонтальной табуляции.

Чтобы строку текста разместить на середине колонки, используется директива .MID, а в качестве аргумента — нужный текст:
.MID ЭТОТ ТЕКСТ ДОЛЖЕН БЫТЬ ПОСЕРЕДИНЕ
В результате получим:

ЭТОТ ТЕКСТ ДОЛЖЕН БЫТЬ ПОСЕРЕДИНЕ

В служебных записках, письмах, при написании эпиграфов текст бывает необходимо расположить по правому краю. Для этого используется директива .RIT (от RIGHT — правый).

Однако, подровняв строки по правому краю, мы получим неровный левый. Избежать этого можно, дополнив строки до определенной длины пробелами и закончив малозаметным символом (скажем, точкой), например так:

```
.RIT Этот текст будет      .
.RIT выровнен по правой    .
.RIT стороне страницы      .
.RIT и иметь ровный        .
.RIT левый край            .
```

12.2. Директивы размещения страниц

Часто, преобразовав текст с помощью программы DOC, мы видим, что он расположился на страницах не очень красиво. Например, нам хотелось бы, чтобы отдельные куски заведомо начинались с новой колонки или страницы. Сделать это просто, если использовать директивы .CLN (от COLUMN — колонка) или .PGN (от PAGE — страница). Директивы аргументов не имеют, распечатываемый после них текст будет соответственно начинаться с новой колонки или страницы. Следует отметить, что если между двумя директивами .PGN или .CLN будут только пустые строки, то эти директивы считаются за одну.

Создав рисунок, указав директивой .SWT, что его не нужно выравнивать по краям, мы с ужасом замечаем, что перевод страницы попал как раз на его середину. Чтобы выйти из этого положения, нужно указать программе DOC, что наш текст-рисунок занимает N строк и его нельзя разрывать посередине. Это делает директива
.TES N

Если мы поставим ее перед рисунком, то DOC проверит, сколько строк осталось на странице, и если рисунок не входит, перейдет на новую колонку или страницу.

Уж если быть привередливыми до конца, то хотелось бы, чтобы на каждой странице стояло название нашей книги. Сделать это просто: достаточно указать подзаголовок (SUBTITLE) директивой

.STL Это моя книга

Текст, используемый в качестве аргумента этой директивы, будет распечатываться на всех последующих страницах. Чтобы изменить его, достаточно дать новую директиву .STL с другим текстом или без него (если нужно совсем отказаться от распечатки подзаголовка).

12.3. Директивы описания структуры

Все описанное до сих пор могут сделать достаточно мощные редакторы текста. Но программа DOC может, кроме того, сама собрать содержание. Чтобы воспользоваться этим, нужно описать структуру нашей книги.

Что входит в понятие структуры? Это разделы и подразделы, пункты и подпункты, аннотация в начале и приложение в конце, а также более мелкие структурные единицы — примечания и перечисления.

Первое, что следует сделать — указать программе DOC, что нужно собирать содержание. В англоязычных книгах содержание называют «TABLE OF CONTENTS»; сокращая это название до трех букв, получим .TOC. Эта директива аргументов не имеет и указывается один раз в том месте текста, куда нужно собирать содержание.

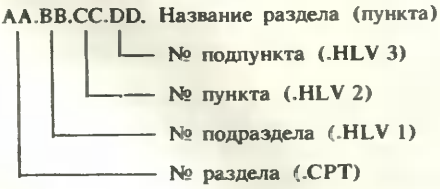
Первый небольшой раздел в книге — аннотация или введение. В ОС РАФОС он назван аннотацией. В RT-11 он назывался INTRODUCTION — введение. Очень простое сокращение .INT, и перед набранным текстом появится заголовок «Аннотация». Директива не имеет аргументов и указывается один раз. Следует отметить, что текст, следующий за директивой .INT, располагается перед содержанием. В тексте можно использовать любые директивы размещения текста. Аннотация заканчивается по директиве, указывающей начало первого раздела.

Разделы — самые крупные структурные единицы книги. Каждый раздел (это может быть, например, глава) имеет номер и название. И если о номере позаботится DOC, то название мы должны указать в качестве аргумента директивы:

.CPT название раздела

DOC разместит указанное нами название посередине строки и снабдит его номером. Некоторые версии DOC каждый раздел начинают с новой страницы.

Более мелкие структурные единицы — подразделы, пункты и подпункты — не имеют персональных директив. Они объединены директивой .HLV с числовым аргументом, который может принимать значения 1 (для подраздела), 2 (для пункта) и 3 (для подпункта). В качестве названия используется текст, следующий за директивой .HLV и заканчивающийся по любой другой директиве. Номера разделов, пунктов и подпунктов будут записываться следующим образом:



Названия разделов и подразделов выводятся в содержание с указанием страницы начала соответствующей структурной единицы.

Случаи, которые мы рассмотрели, являются самыми обычными и простыми. Но есть два случая, которые требуют либо специальной директивы, либо какого-нибудь приема.

Во-первых, может возникнуть необходимость пронумеровать абзац, не давая ему названия и не вынося этот номер в содержание. Для этого используется комбинация из директивы .HLV с соответствующим числовым аргументом, после которой следует текст абзаца; последней ставится директива .LIN.

Во-вторых, может возникнуть обратная ситуация: необходимость вывести в содержание строку или группу строк, но так, чтобы в отформатированном тексте этой строки не осталось. Нужно понимать, что если строка выводится между какими-либо строками в содержании, то директива с текстовым аргументом должна располагаться в любом месте текста между соответствующими директивами .HLV или .CPT. Название директивы, по-видимому, образовалось из фразы TEXT COPY TO TABLE, и в сокращенном виде она записывается так: .TCT дополнительно выводимый в содержание текст

Если после директивы .HLV или .CPT стоит директива .TCT, то в содержании перед выводимым текстом будет пропущена пустая строка. Если директивы .TCT идут подряд, то этого не происходит.

Можно отметить и более мелкие структурные единицы — различного рода перечисления и уточнения, которые нет смысла

нумеровать и выводить в содержание, но необходимо как-то выделить в тексте.

Начнем с простого перечисления. При его форматировании левый край текста немного сдвигается вправо и в начале такого абзаца ставится дефис. В качестве примера перечислим все мелкие структурные единицы:

— простое перечисление — когда каждый абзац выделяется только дефисом в начале первой строки;

— определение — когда в одной строчке записывается термин, а в следующих — его определение так же, как в простом перечислении;

— примечание — когда для выделения в начале абзаца пишется слово «Примечание», а дальнейший текст форматируется обычным образом.

А перед обработкой программой DOC этот текст выглядел так:

```
.AND 0
простое перечисление — когда каждый абзац выделяется только дефисом в начале первой строки;
.AND 0
определение — когда в одной строчке записывается термин, а в следующих — его определение так же, как в простом перечислении;
.AND 0
примечание — когда для выделения в начале абзаца пишется слово «Примечание», а дальнейший текст форматируется обычным образом. Как видно, простое перечисление указывается с помощью директивы
.AND.
```

Определение записывается одной директивой .TBL, которая по сути является составной:

```
.TBL Термин                    .BLN
Текст определения              Термин
                               .AND 0
                               Текст определения
```

Вот пример ее использования:

```
.TBL AND 0
директива используется для выделения перечисления;
.TBL .TBL Текст
директива используется для выделения определения некоторого термина, который записывается в качестве аргумента директивы.
```

А таким будет результат форматирования:

```
.AND 0
    — директива используется для выделения перечислений;
.TBL    Текст
    — директива используется для выделения определения некоторого термина, который записывается в качестве аргумента директивы.
```

Последняя директива этой группы указывает на то, что далее следует примечание. Директива называется .ATT/, аргументов не имеет, единственное ее действие — напечатать в начале абзаца слово «Примечание». Ее можно заменить следующим образом:

```
.LIN 1
Примечание:
(далее текст примечания)
```

Конец примечания лучше всего указывать явно словами «Конец примечания».

Заканчиваться книга может приложением некоторого справочного материала. В нем следует напечатать: в начале страницы слово «Приложение» и его номер, а посередине следующей строки — название приложения. Это делает директива

```
.APN название приложения
```

За номером проследит программа DOC.

После директивы .APN можно использовать любые директивы размещения текста. Следует отметить, что эта директива обнуляет счетчик разделов и подразделов, поэтому если после нее поставить директиву .CPT, то в содержании этот раздел будет иметь номер 1.

12.4. Служебные директивы

Служебные директивы программы DOC не имеют непосредственного отношения к тексту, но позволяют более профессионально управлять и распечаткой текста, и файлами.

Первое, что требуется от профессионала,— аккуратность. В каждом файле, даже если это простой текст, а не программа, необходимо оставить след — когда и кто этот файл создал, когда, кто и какие изменения в него вносил. Конечно, не следует выводить эту информацию в распечатываемый текст. Такая возможность обеспечивается директивой .COM (COMMENT — коментарий); текст комментария записывается в качестве аргумента. Например, вот так:

```
.COM этот текст является комментарием
```

Программа DOC во время своей работы выводит эти строки на экран терминала, поэтому комментарии можно использовать как признаки, какая часть текста в данный момент обрабатывается.

Текст, полученный с помощью DOC, напечатан очень плотно, если сравнивать с пишущей машинкой — через один интервал. Для экономии бумаги это хорошо, но такой текст неудобно читать и редактировать; статьи в журнал, например, нужно печатать через строчку (через два интервала). Директива .JMP (от JUMP — прыжок) позволяет при форматировании перескакивать через указанное в ее аргументе число строк. Эта директива не влияет на аргументы

остальных директив и может указываться в любом месте текста.

Директива .MRR устанавливает длину строки и имеет три числовых аргумента, которые задают:

первый — длину строки для директив .LIN, .BLN, .SWT, .APN, .MID, .RIT, .CPT; по умолчанию для 11-го формата — 60, для 12-го формата — 58;

второй зарезервирован для дальнейшего использования; по умолчанию для 11-го формата — 44, для 12-го — 44;

третий — длину строки для директив .AND и .TBL; по умолчанию для 11-го формата — 51, для 12-го — 49.

Директива .MRL задает номер позиции, с которой должна начинаться строка, и имеет три аргумента, указывающих:

первый — левую границу для директив .LIN, .BLN, .SWT, .APN, .MID, .CPT; по умолчанию — 1;

второй зарезервирован для дальнейшего использования; по умолчанию — 12;

третий — левую границу для директив .AND и .TBL; по умолчанию — 10.

12.5. Работа с программой DOC

Когда текст с директивами программы DOC набран, наступает время работать самой программе. Во второй и третьей версии RT-11 DOC поддерживалась командой клавиатурного монитора DOCUMENT. Начиная с версии V5.0 для вызова можно использовать имя программы (DOC) или вызывать ее с помощью команды RUN. Обязательно нужно сказать, какой файл обрабатывается:

.DOC DX1:ROMAN.DOC

и отформатированный текст начинает выводиться на печатающее устройство. А точнее, DOC выводит отформатированный текст в файл на устройство DPR: (DOCUMENT PRINTER) с именем первого входного файла и типом .DPR. По умолчанию устройством DPR: является печатающее устройство.

12.6. Сообщения об ошибках

Программа DOC написана в основном на языке Фортран, поэтому расшифровку сообщений об ошибках следует искать в описании транслятора Фортрана.

Г. ГУТМАН
г. Куйбышев

Реализация рекурсивных алгоритмов на Бейсике

Несмотря на то что рекурсивные алгоритмы упоминаются в учебнике информатики, рекурсия для большинства школьников остается «таинственной незнакомкой». Еще бы! Примеров в учебнике явно недостаточно, да и те допускают очевидное решение с помощью обычного цикла. В то же время есть несколько классических задач, прекрасно иллюстрирующих возможности рекурсии. Это задачи о перестановках, размещении ферзей на шахматной доске, поиске пути в лабиринте, игра «Ханойские башни» и т. д.

Другим моментом, препятствующим изучению понятия рекурсии, является отсутствие соответствующего аппарата в ряде распространенных учебных языков, в частности в Бейсике. Однако, как мы увидим дальше, на Бейсике можно достаточно просто реализовать рекурсивные алгоритмы. Возможно, у читателя возникнет вопрос: зачем нужна «самодельная» рекурсия? Во-первых, часто рекурсивный алгоритм оказывается короче и проще итеративного. Во-вторых, самостоятельная реализация рекурсивного алгоритма позволяет глубже разобраться и в обычной конструкции — подпрограмме.

Подпрограмма в Бейсике есть часть общей программы. Обращение к ней производится оператором GOSUB ⟨номер строки⟩. При этом в стеке возврата запоминается адрес начала следующего оператора. Выход из подпрограммы происходит по оператору RETURN, причем в качестве адреса перехода берется последнее значение из стека возврата. Покажем, что этот механизм можно использовать и для организации рекурсивной подпрограммы.

Рассмотрим следующий пример:

```
10 REM======== ПРИМЕР РЕКУРСИИ =======
20 K=1
30 GOSUB 100
40 END
100 REM-------------------------------
110 REM РЕКУРСИВНАЯ ПОДПРОГРАММА
120 REM-------------------------------
```

```
130 IF K=5 GOTO 170
140     PRINT K
150     K=K+1
160     GOSUB 100
170 K=K-1
180 PRINT K:
190 RETURN
200 REM=========================
```

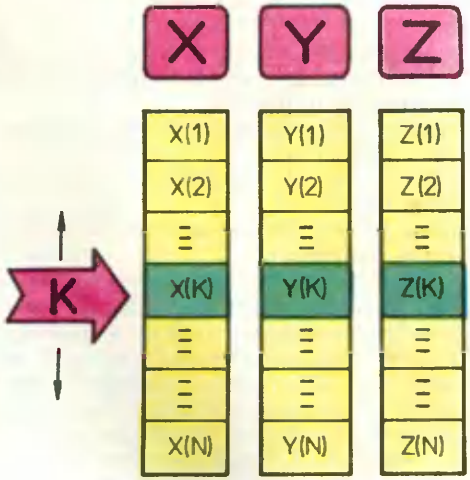
При первом обращении к рекурсивной подпрограмме из строки 30 в стек возврата будет записан адрес строки 40, при втором обращении — из строки 160 — адрес следующей строки (170) и т. д. Проследим за стеком возврата.

K	Стек возврата				
1	40				
2	170	40			
3	170	170	40		
4	170	170	170	40	
5	170	170	170	170	40
4	[170]	170	170	170	40
3	[170]	170	170	40	
2	[170]	170	40		
1	[170]	40			
0	[40]	стек	пуст		

Адреса для возврата по оператору RETURN берутся из стека в обратном порядке (в таблице они выделены рамками). В последнюю очередь будет выполнен возврат к строке 40 и останов по оператору END.

Структура примера характерна для записи рекурсивных программ на Бейсике. Строки 10—40 — это основная программа, в которой задаются начальные значения параметров и/или переменных рекурсивной подпрограммы, производится первое обращение к ней и останов после ее выполнения. Строки 100—200 — это рекурсивная подпрограмма. Вывод параметра K позволяет проследить за переходами с одного уровня рекурсии на другой: он увеличивается на единицу перед каждым обращением к подпрограмме и уменьшается на единицу перед выходом из нее. Ниже мы встретимся с другим вариантом, при котором значение K уменьшается при «погружении» в рекурсивную подпрограмму и увеличивается при «всплытии».

Главной проблемой при организации рекурсии является сохранение результатов работы рекурсивной подпрограммы при последующих самообращениях. В языках, допускающих рекурсию, при входе в рекурсивную подпрограмму создаются новые экземпляры используемых переменных. Они «закрывают своим телом» прежние значения. При выходе из рекурсивной подпрограммы эти экземпляры уничтожаются и становятся



доступными предыдущие значения.

Для реализации этого механизма на Бейсике приходится хранить значения переменных на всех уровнях рекурсии, т. е. каждую простую переменную нужно заменить таблицей. Эту идею поясняет рис. 1. В данный момент программе доступны только X(K), Y(K), Z(K). При сдвиге указателя K вверх «откроются» X(K—1), Y(K—1), Z(K—1), при сдвиге вниз — X(K+1), Y(K+1), Z(K+1).

Теперь мы имеем достаточно средств для реализации рекурсии на Бейсике.

Рассмотрим, например, рекурсивный алгоритм решения головоломки «Ханойские башни». Напомним, что в этой головоломке имеются три стержня, на одном из которых собрано N уменьшающихся кружков. Требуется переложить их на третий стержень, пользуясь вторым как вспомогательным, по определенным правилам (перекладывать по одному кружку и не класть больший кружок на меньший).

Обозначим X — начальный стержень; Y — промежуточный стержень; Z — конечный стержень.

Алгоритм решения этой задачи известен:

```
алг ХАНОЙСКИЕ БАШНИ (нат N)
  арг
нач
  ПЕРЕЛОЖИТЬ (N, 1, 2, 3)
кон
алг ПЕРЕЛОЖИТЬ (цел K, X, Y, Z)
  арг K, X, Y, Z
нач если K>0
       то ПЕРЕЛОЖИТЬ (K—1, X, Z, Y)
          вывод «диск» K «переложить с»
          X «на» Z
          ПЕРЕЛОЖИТЬ (K—1, Y, X, Z)
       все
кон (возврат)
```

При первом обращении к рекурсивной подпрограмме ПЕРЕЛОЖИТЬ (K, X, Y, Z) ее аргументы получают значения K=N, X=1, Y=2, Z=3. Проследим за выполнением подпрограммы по шагам:

Шаг 1: переложить диск N—1 с X=1 на Y=2 стержень.

Шаг 2: диск N переложить с X=1 на Z=3 стержень.

Шаг 3: переложить диск N—1 с Y=2 на Z=3 стержень.

В свою очередь, выполнение шага 1 предполагает выполнение такого же трехшагового алгоритма, однако при этом стержень Y используется как конечный (вместо Z), а стержень Z — как промежуточный. То же относится и к выполнению шага 3, только стержень Y становится начальным (вместо X), а стержень X — промежуточным. Таким образом, на каждом уровне рекурсии параметры X, Y и Z получают новые значения в результате перенумерации стержней.

При реализации на Бейсике простые переменные X, Y, Z заменяются таблицами. В остальном нижеприведенная программа является «дословным» переводом рекурсивного алгоритма.

```
10 REM======= ХАНОЙСКИЕ БАШНИ =======
20 INPUT N
30 DIM X(N), Y(N), Z(N)
40 X(N)=1: Y(N)=2:Z(N)=3
50 K=N: GOSUB 100
60 END
100 REM------------------------------
110 REM РЕКУРСИВНАЯ ПОДПРОГРАММА
120 REM------------------------------
130 IF K=0 GOTO 190
140    X(K-1)=X(K)
141    Y(K-1)=Z(K)
142    Z(K-1)=Y(K)
150      K=K-1: GOSUB 100
160    PRINT "ДИСК"K" - С "X(K)" НА " Z(K)
170    X(K-1)=Y(K)
171    Y(K-1)=X(K)
172    Z(K-1)=Z(K)
180      K=K-1: GOSUB 100
190 K=K+1: RETURN
200 REM==========================
```

Проследим за работой программы при N=3. В основной программе (строки 10—60) выполняется начальное присваивание X(3)=1, Y(3)=2, Z(3)=3, K=3 и происходит первое обращение к рекурсивной подпрограмме оператором GOSUB 100. Результаты работы подпрограммы оформим в виде таблицы.

При дальнейшей работе подпрограммы диски 1 и 2 со второго стержня будут перенесены на третий стержень на диск 3. Читатель может проверить это самостоятельно.

Рекурсию в цикле можно рассматривать как обобщение понятия вложенных циклов. Рассмотрим следующую задачу: получить все последовательности из чисел —1, 0, 1 длины 4. Обычный алгоритм будет содержать конструкцию

для a_1 **от** —1 **до** 1
нц для a_2 **от** —1 **до** 1
нц для a_3 **от** —1 **до** 1
нц для a_4 **от** — 1 **до** 1

Номера строк	Значения переменных K	X(K)	Y(K)	Z(K)	Стек возраста			
40—50	3	1	2	3	60			
140—150	2	1	3	2	160	60		
140—150	1	1	2	3	160	160	60	
140—150	0				160	160	160	60
190	1	1	2	3	[160]	160	160	60
160	ДИСК 1 ПЕРЕЛОЖИТЬ С 1 НА 3							
170—180	0				190	160	160	60
190	1				[190]	160	160	60
190	2	1	3	2	[160]	160	60	
160	ДИСК 2 ПЕРЕЛОЖИТЬ С 1 НА 2							
170—180	1	3	1	2	190	160	60	
140—150	0				160	190	160	60
190	1	3	1	2	[160]	190	160	60
160	ДИСК 1 ПЕРЕЛОЖИТЬ С 3 НА 2							
170—180	0				190	190	160	60
190	1				[190]	190	160	60
190	2				[190]	160	60	
190	3	1	2	3	[160]	60		
160	ДИСК 3 ПЕРЕЛОЖИТЬ С 1 НА 3							

и так далее

```
            нц вывод a1, a2, a3, a4
            кц
        кц
    кц
кц
```

Недостаток этого алгоритма очевиден — он генерирует только последовательности длины 4, т. е. годится только для решения поставленной задачи. Для построения последовательностей длины N структура вложенных циклов в принципе не пригодна. Решение общей задачи можно получить другими способами, в том числе используя рекурсию в цикле вместо вложенных циклов.

Основная идея рекурсивного алгоритма: положим первый элемент последовательности $a_1=-1$ и «приставим» к нему *все* последовательности на единицу меньшей длины. Получим часть решения исходной задачи. Повторив те же действия для $a_1=0$ и $a_1=1$, получим полное решение задачи.

Напишем «заготовку» рекурсивной подпрограммы:

```
алг ПЕРЕБОР
нач для aK от — 1 до 1
    нц K:=K+1
       ПЕРЕБОР
    кц
K:=K—1
кон (возврат)
```

Задав перед обращением к алгоритму K=1, получим бесконечную последовательность $a_1=-1$, $a_2=-1$, $a_3=-1$, ...!? Очевидно, что при K=N+1 алгоритм должен вывести полученную последовательность (для этого переменные a_1, a_2, ... a_N следует рассматривать как элементы таблицы *a* [1:N]) и вернуться для продолжения работы на N-й уровень рекурсии. Окончательный алгоритм примет вид

```
алг ПОСЛЕДОВАТЕЛЬНОСТИ (нат N)
    арг N
нач цел таб a [1:N]
    K:=1
    ПЕРЕБОР_1
кон
алг ПЕРЕБОР_1
нач нат I
    если K= N+1
      то для I от 1 до N
         нц вывод a [I]
         кц
      иначе для a[K] от —1 до 1
            нц K:=K+1
               ПЕРЕБОР_1
            кц
    все
    K:=K—1
кон (возврат)
```

При переводе данного алгоритма на Бейсик выяснилось, что имеющийся в распоряжении автора вариант языка не разрешает в качестве параметра цикла FOR — NEXT использовать индексированные переменные, поэтому в программе используется конструкция, эквивалентная циклу «пока».

```
10 REM====== ПОСЛЕДОВАТЕЛЬНОСТИ =====
20 INPUT N
30 DIM A(N)
40 K=1: GOSUB 100
50 END
100 REM------------------------------
110 REM РЕКУРСИВНАЯ ПОДПРОГРАММА
120 REM------------------------------
130 IF K<=N GOTO 180
140    FOR I=1 TO N
150       PRINT A(I);
160    NEXT I: PRINT
170    GOTO 220
180 A(K)=-1
190 IF A(K)>1 GOTO 220
200    K=K+1: GOSUB 100
210    A(K)=A(K)+1: GOTO 190
220 K=K-1: RETURN
230 REM==========================
```

Модифицируя эту программу, можно получить последовательности из любых чисел, как заранее известных, так и вводимых с клавиатуры по оператору INPUT.

С. ГВОЗДЕВ, Г. ЭРНЕСТСОНС

Использование плавающей арифметики Фокала в БК-0010

Компьютер «Электроника БК-0010» (в дальнейшем просто БК), получивший большое распространение, оснащался до последнего времени интерпретатором языка Фокал [1], выполненным в микросхеме К1801РЕ2-018 (или К1801РЕ2-084 в комплекте с классом КУВТ-86). С помощью Фокала можно создавать разнообразные программы как чисто вычислительного, так и игрового характера, используя графические возможности и доступ к порту БК.

Однако использование только Фокала иногда препятствует эффективному решению задачи на ЭВМ. Это особенно проявляется при программировании динамических графических изображений или при управлении внешними устройствами. Некоторые задачи вообще невозможно решить, оставаясь в рамках Фокала, например создание алфавита пользователя или программирование звуковых эффектов.

Применение подпрограмм пользователя в рамках так называемого FOCODa (автор В. Н. Андреев, Москва) также не всегда дает желаемый результат. Поэтому предпочтительнее программировать исключительно в кодах, полностью отказываясь от Фокала, получая в качестве компенсации в свое распоряжение все возможности БК.

Программирование в кодах позволяет не только повысить скорость исполнения программы, но и сэкономить память, тем более, что этот ресурс в БК крайне ограничен.

Однако, отказываясь от Фокала, приходится считаться с отсутствием в системном программном обеспечении БК арифметики чисел с плавающей запятой и библиотеки подпрограмм для вычисления стандартных функций.

В статье предлагается простой метод доступа на уровне кодов БК к находящимся в ПЗУ Фокал-арифметике и встроенным функциям Фокала. В этом случае удается сэкономить память, затраты которой были бы значительными, если в нее вводить собственный вариант подпрограмм для работы с числами с плавающей запятой. Так, объем программ в ПЗУ, реализующих Фокал-арифметику, составляет 2,7К байт.

Как уже отмечалось выше, программирование в кодах позволяет получить выигрыш при решении задачи относительно ее реализации на Фокале. Для иллюстрации рассмотрим пример, часто встречающийся при сборе данных. Предположим, что некоторое устройство, например измеритель температуры, подключено к порту БК и выдает на него по приказу ЭВМ двоичные числа, пропорциональные значению температуры. Таким приказом является установка младшего бита порта на короткое время в единицу. Пусть необходимо произвести 1000 измерений и поместить их значения в массив T (i). Программа на Фокале, производящая указанные действия, будет выглядеть следующим образом:

```
10.10 X FP (1,177777); C Очистка регистра вывода
10.20 S N=1000; C Установка числа измерений
10.30 F I=1,N; X FP (2,1); S T(I)=FP(0,177777)
10.40 Q
```

В этой программе около 95 % машинного времени уходит на ее интерпретацию. Соответствующая программа в машинных кодах работает в 40—50 раз быстрее. Кроме увеличения скорости получается выигрыш в памяти, ведь для хранения числа в Фокале отводится 8 байтов, в то время как фактически оно размещается в 4 байтах. Таким образом, выигрыш при хранении данных равняется 4 байтам на каждое число.

Отметим, что выигрыш в скорости тем значительнее, достигая 2—3 порядков, чем более просты математические операции, используемые в Фокал-программе. Однако даже при вычислении сложных функций типа FSIN(X) программа в кодах работает в 1,5 раза быстрее.

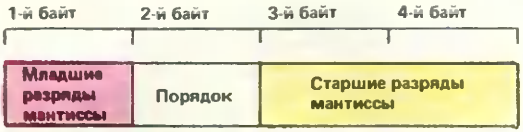
Программу в кодах, аналогичную вышеописанной фокальной, мы приведем после разбора арифметики чисел с плавающей запятой.

Фокал-арифметика

В связи с недостаточностью в публикациях сведений о Фокал-арифметике дадим краткий обзор на эту тему. Будем считать, что читателю известны архитектура БК и элементы программирования на ассемблере (см., например, [2]).

В Фокале реализована арифметика вещественных чисел с плавающей запятой одинар-

ной точности. В памяти ЭВМ числа представлены в двусловном формате. Порядок записывается в одном байте, а мантисса занимает три байта. Отрицательные порядок и мантисса кодируются в дополнительном коде.



Арифметические операции с числами производятся в трехсловном (слово — 2 байта) сумматоре, реализованном программно и размещенном в памяти БК в адресах 1710, 1712, 1714.

Кодирование операций осуществляется с помощью псевдокоманд, при встрече которых в тексте программы происходит прерывание по вектору 10 (прерывание по коду резервной команды) и управление передается на обработку этих команд.

Арифметические операции в Фокале можно условно разделить на двухместные (сложение, умножение) и одноместные (целая часть, абсолютная величина).

Первым операндом в двухместных операциях является содержимое сумматора, второй операнд выбирается из памяти в зависимости от метода адресации. Результат операции всегда помещается в сумматор.

Двухместная операция кодируется четырехзначным восьмеричным числом 70XY, где

X — код операции;

Y — метод адресации операнда (табл. 1, 2).

При описании арифметических операций и функций используются следующие обозначения: S — сумматор, Op — операнд.

Таблица 1

Код X	Операция	Мнемоника
0	op -> S	FGET
1	S+op -> S	FADD
2	S-op -> S	FSUB
3	S/op -> S	FDIV
4	S*op -> S	FMUL
5	S^op -> S	FPOW
6	S -> op	FPUT

Таблица 2

Код У	Адрес операнда	Мнемоника
0	Адрес в следующем слове	@#op
1	Адрес в R2	(R2)
2	(R2)+4, адрес в R2	+(R2)
3	Операнд в вершине стека	(SP)
4	Адрес в вершине стека	@(SP)
5	Операнд в следующих двух словах	#op
6	Не работает	

Типы адресации достаточно очевидны и ясны из обозначений табл. 2. Однако есть нюансы относительно традиционных методов адресации в архитектуре ЭВМ. Так, использование адресации с кодом 2 (автоинкрементная адресация) имеет ту особенность, что изменение содержимого регистра R2 производится до операции, а не после, как обычно при выполнении машинных команд. При размещении операнда в стеке (адресация с кодом 3) программист сам должен следить за указателем стека.

Псевдокоманды с кодом 707Y задают одноместные операции, выполняющиеся над содержимым сумматора (табл. 3).

Таблица

Команда	Операция	Мнемоника
7070	Нормализация числа в s	FNOR
7071	Целая часть S→R1	FINT
7072	Sgn(S)→S	FSGN
7073	Abs(S)→S	FABS
7074	—S→S	FNEG
7075	Перевод числа, заданного символьной строкой в S	FREAD
7076	Вывод S на экран R2 — формат вывода.	FPRINT
7077	0→S	FZERO

При выполнении псевдокоманд содержимое регистров сохраняется, кроме тех случаев, когда изменение содержимого регистров обусловлено самой псевдокомандой. Необходимо отметить, что признаки NZVC после выполнения псевдокоманд не устанавливаются в соответствии с результатом операции.

Проиллюстрируем на примерах использование описанных операций. Программы в кодах БК приводятся в соответствии с мнемоникой, принятой в ассемблере. В приводимых примерах значащими являются только столбцы с адресом памяти и содержимым слова по данному адресу. Поля с мнемоникой команд и комментарием приводятся только в целях облегчения понимания.

Загрузка сумматора некоторым числовым значением производится с помощью операции FGET:

```
Адрес   Содержимое  Мнемоника       Комментарий
002000  007000        FGET @#A     ; Операция  FGET
002002  002100                   ; Адрес операнда
. . . . . . . . . . . . . . . . . . . . . . . . . . .
002100  000002 A:                ; Число 2.0 в формате
002102  040000                   ; с плавающей запятой
```

Та же самая операция возможна с адресацией через регистр R2.

```
002000  012702      MOV #A,R2    ; Загружаем адрес
002002  002100                   ; переменной  A  в R2
002004  007001      FGET (R2)    ;A в сумматор
```

Адресация через регистр R2 особенно эффективна, когда переменные занимают смежные участки памяти БК. В этом случае удобно использовать автоматическую модификацию индексного регистра, так называемую автоиндексацию. Рассмотрим пример, в котором применяется автоиндексация. Пусть у нас есть фрагмент Фокал-программы:

```
10.10 S C=2.0+3.0
10.20 T C; Q
```

Программа в кодах может выглядеть следующим образом:

```
002000  012702      MOV #(A-4),R2  ;Адрес массива перемен-
002002  002014                     ;ных с учетом автомати-
                                   ;ческого наращивания.
002004  007002      FGET +(R2)     ;Занести в сумматор 2.0
002006  007012      FADD +(R2)     ;Добавить к сумматору 3.0
002010  007062      FPUT +(R2)     ;Результат запомнить в C
002012  005002      CLR R2         ;Подготовиться к выводу
002014  007076      FPRINT         ;Вывести сумматор на экран
002016  000000      HALT
002020  000002 A:                 ;2.0
002022  040000
002024  000002 B:                 ;3.0
002026  060000
002030          C:                 ;Здесь будет результат
```

В приведенном примере в переменных A и B записаны константы в формате чисел с плавающей запятой. В данном случае перевод в нужный вид был сделан вручную. Иногда такое преобразование можно поручить БК. Очевидно, что в процессе решения задач возникают ситуации, требующие преобразования чисел из одного вида в другой. Например, целые числа при программировании в кодах гораздо удобнее. Но, перед тем как обратиться к Фокал-арифметике, целое число надо преобразовать в нормальный двусловный формат. Целое число, расположенное по адресу A, можно поместить в сумматор при помощи следующего фрагмента программы:

```
002000  012737      MOV #17,@#1710 ;Порядок целого числа
002002  000017                     ;равен I5
002004  001710
002006  013737      MOV @#A,@#1712 ;Записали целое число
002010  002100                     ;в виде
002012  001712                     ;ненормализованной мантиссы
002014  005037      CLR @#1714     ;Незначащие разряды в 0
002016  001714
002020  007070      FNOR           ;Нормализовали сумматор
```

В дальнейшем значение сумматора можно или использовать непосредственно в вычислениях, или записать в память БК.

В практике часто приходится вводить числовые данные с клавиатуры БК. При этом в памяти образуется последовательность байтов, которой кодируется число в символьном виде в коде КОИ-8. Для перевода числа в двоичный вид можно воспользоваться операцией FREAD. Символьная строка предварительно должна быть перекодирована, так как в Фокале для некоторых символов используются коды, отличные от стандартных кодов КОИ-8. При записи числа, в частности, необходимы символы «—» (минус) и «+» (плюс). В Фокале для этих символов употребляются восьмеричные коды 202 и 201 вместо 55 и 53. Признаком конца символьной строки является код 217. В следующем примере с клавиатуры вводится число, коды клавиш +, — и **ВВОД** перекодируются. Затем число переводится в плавающий формат и записывается в сумматор.


```
002000  012701        MOV #A,R1      ;Поместить адрес
002002  002100                       ;символьной строки в RI
002004  104016 4¤:    EMT 16         ;Ввести код с клавиатуры в R0
002006  120027        CMPB R0,#'+    ;Введен "+"?
002010  000053
002012  001003        BNE 1¤         ;Нет
002014  112700        MOVB #201,R0   ;Да. Заменить код
002016  000201
002020  000413        BR 2¤
002022  120027 1¤:    CMPB R0,#'-    ;Введен "-"?
002024  000055
002026  001003        BNE 3¤         ;Нет
002030  112700        MOVB #202,R0   ;Да
002032  000202
002034  000405        BR 2¤
002036  120027 3¤:    CMPB R0,#12    ;Нажат "ВВОД"?
002040  000012
002042  001002        BNE 2¤         ;Нет
002044  112700        MOVB #217,R0   ;Код завершения строки
002046  000217
002050  110021 2¤:    MOVB R0,(R1)+  ;Записали символ в буфер
002052  120027        CMPB R0,#217   ;Если закончили ввод,
002054  000217                       ;то выполнить перевод
002056  001352        BNE 4¤         ;Иначе ввести следующий символ
002060  012703        MOV #A,R3      ;Для перевода надо поместить
002062  002100                       ;адрес символьной строки в R3
002064  112304        MOVB (R3)+,R4  ;Обязательно поместить
                                     ;первый символ в R4
002066  112737        MOVB #20,@#1723 ;Обязательно заполнить
```

```
002070  000020
002072  001723
002074  007075      FREAD           ;Поместить число в сумматор
002076  000000      HALT
002100          A:                  ;Буфер ввода
```

Если символьная строка не требует перекодировки, то для перевода числа в плавающий формат необходимы строки программы, записанные в адресах с 2060 по 2074.

Операция FREAD вводит в сумматор всегда положительное число. Если символьная строка задает отрицательное число, то после операции FREAD устанавливается разряд условия N. В этом случае можно воспользоваться операцией FNEG для правильного представления числа в сумматоре.

В списке операций Фокала отсутствует операция для перевода числа из плавающей формы в символьный вид. Однако операция FPRINT позволяет вывести на экран содержимое сумматора. Регистр R2 содержит при этом требуемый формат. Восьмеричная константа 4012 соответствует формату Фокала %8.04. Если R2 равен нулю, то число будет выведено в экспоненциальной форме. При использовании FPRINT необходимо иметь в виду, что операция не сохраняет содержимое сумматора.

Завершим рассмотрение операций Фокала фрагментом программы в кодах БК, соответствующей Фокал-программе, которая приведена в начале статьи.


```
002000  012705      MOV #1000.,R5   ;Число измерений 1000
002002  001750
002004  012702      MOV #<T-4>,R2   ;Адрес массива измерений
002006  002052
002010  012701      MOV #177714,R1  ;Адрес порта ввода-вывода
002012  177714
002014  005011      CLR (R1)        ;Очистить регистр вывода
002016  052711  1$: BIS #1,(R1)     ;Произвести измерение
002020  000001
002022  042711      BIC #1,R1
002024  000001
002026  012737      MOV #17,@#1710
002030  000017
002032  001710
002034  011137      MOV (R1),@#1712 ;Записать измерение
002036  001712                      ;в сумматор
002040  005037      CLR @#1714
002042  001714
002044  007070      FNOR            ;Перевести в плавающий формат
002046  007062      FPUT +(R2)      ;Записать в массив T
002050  077516      SOB R5,1$       ;Следующее измерение
002052  000000      HALT
002054          T:                  ;Массив измерений T
```

Данная программа иллюстрирует еще один источник экономии памяти при программировании в кодах. Несмотря на довольно длинную запись, программа в кодах занимает в памяти ЭВМ примерно в два раза меньше места, чем ее прототип на Фокале. В среднем, как показывает опыт, данная пропорция сохраняется для достаточно длинных и содержательных программ.

Встроенные математические функции Фокала

Обращение к встроенным функциям Фокала производится как к обычным подпрограм-

мам, в качестве регистра возврата используется счетчик команд PC (или R7), т. е.

JSR PC, ⟨адрес функции в ПЗУ Фокала⟩

В табл. 4 приведены адреса встроенных функций для микросхем К1810РЕ2-018 и К1810РЕ2-084.

Таблица 4

Адрес для микросхемы "018"	Функция	Мнемоника	Адрес для микросхемы "084"
126550	int(S) ->S	FITR	126672
132260	cos(S) ->S	FCOS	132262
132250	sin(S) ->S	FSIN	132252
132122	S ->S	FSQT	132124
132550	Случайное	FRAN	132552
	число ->S		126666
126554	sign(S) ->S	FSGN	126662
126540	|S| ->S	FABS	134134
134132	lg(S) ->S	FLOG10	134160
134156	ln(S) ->S	FLOG	134420
134416	exp(S) ->S	FEXP	133704
133702	tg(S) ->S	FTAN	134634
134632	arctg(S) ->S	FATAN	133660
133656	arccos(S) ->S	FACOS	133404
133402	arcsin(S) ->S	FASIN	

При использовании встроенных функций программист должен сам побеспокоиться о сохранении регистров ЭВМ.

Программа начальной инициализации

Опишем действия, которые надо проделать, прежде чем воспользоваться Фокал-арифметикой или встроенными функциями. Связь с программным обеспечением, находящимся в ПЗУ, осуществляется через вектора прерывания. Выше уже упоминалось, что при встрече кодов псевдокоманд происходит прерывание по резервной команде (вектор прерывания 10). При обработке псевдокоманд Фокал использует внутренние обслуживающие подпрограммы. Служебные подпрограммы выполняют такие действия, как работа со строками, ввод-вывод символьной информации, вычислительные действия, выдача сообщений об ошибках. Часть служебных подпрограмм вызывается с помощью TRAP-запросов. Вектор прерывания для TRAP-запросов содержится в ячейке 34.

Следовательно, для успешной работы с программным обеспечением, размещенным в ПЗУ Фокала, необходимо создать соответствующую рабочую среду. Если программа пользователя написана без единой ошибки, то для правильной обработки псевдокоманд достаточно перед их выполнением заменить содержимое ячеек 10 и 34 на адреса связи с Фокалом 125636 и 121136 соответственно. Однако в этом случае программа пользователя является беззащитной перед любыми ненормальными ситуациями, происходящими при работе обслуживающих подпрограмм Фокала. Например, невозможно гарантировать, что в процессе решения задачи не случится попытка деления на нуль или попытка извлечь корень из отрицательного числа. В этом случае (и при любой другой ошибке) управление передается Фокалу, который в наказание стирает как программу пользователя, так и все данные.

Предлагаемая вспомогательная программа осуществляет корректный интерфейс между программой пользователя в кодах БК, обслуживающими подпрограммами Фокала и монитором, из которого был произведен запуск программы пользователя. В любом случае вход в программу пользователя происходит из вспомогательной программы, которую мы будем называть в дальнейшем программой начальной инициализации ПИФ.

Основными функциями ПИФ являются следующие:

сохранение области векторов прерывания монитора, из которого производится запуск программы;

создание необходимой рабочей среды для Фокала;
перехват всех TRAP-запросов и анализ их корректности;
обработка ошибочных TRAP-запросов;
возврат в монитор, из которого был произведен запуск программы.

Область памяти 1000—1777 используется программой ПИФ и рабочими ячейками Фокала. Приведем распределение адресного пространства.

1000 Стартовый адрес программы. Сохранение области векторов прерывания, заполнение рабочих ячеек Фокала, установка векторов прерывания, установка указателя стека.
1060 «Ловушка» для TRAP-запросов.
1120 Обработка недействительных TRAP-запросов.
1142 Выход в вызывающий монитор: восстановление векторов прерывания.
1160 Свободная область.
1200 Адреса обработки TRAP-запросов с 200 по 262.
1264 Резерв для использования TRAP с 264 по 376.
1400 Подпрограмма для сохранения и восстановления векторов прерывания, область сохранения.
1500 Рабочие ячейки Фокала.
2000 Первая команда программы пользователя.

Программа пользователя в кодах должна размещаться начиная с адреса 2000. Более того, по адресу 2000 должна располагаться первая исполняемая команда программы. Однако пуск программы производится через ПИФ, т. е. с адреса 1000. Завершаться программа пользователя может традиционной командой HALT. Естественно, результат при нажатии на клавишу **СТОП** будет точно такой же.

Выше речь уже шла об ошибочных ситуациях при исполнении операций Фокала или при вычислении стандартных функций. В этом случае ПИФ обеспечивает нормальный останов программы с выдачей дополнительно некоторого диагностического сообщения. Например, если произошло деление на нуль, то на экран выдается:

ПОПЫТКА ДЕЛЕНИЯ НА НУЛЬ

Четные номера TRAP-запросов с кодами 264—376 являются свободными и могут быть использованы программистом для обращения к своим подпрограммам с помощью программного прерывания TRAP. Область памяти с адресами 1264—1376 зарезервирована для соответствующих адресов подпрограмм пользователя. Если подпрограмму, расположенную начиная с адреса SUBPR, пользователь захочет вызывать с помощью, например, запроса TRAP 264, то предварительно необходимо адрес подпрограммы поместить в слово с адресом 1264. Возврат из подпрограммы производится через регистр R5.

```
        MOV #SUBPR,@#1264   ;Адрес подпрограммы
                            ;пользователя - в
                            ;таблицу TRAP-запросов
. . . . . .
        TRAP 264            ;Вызов подпрограммы
. . . . . .
SUBPR:                      ;Начало подпрограммы
        RTS R5              ;Возврат
```

Кроме того, программист может включать в свою программу и некоторые служебные TRAP-запросы Фокала, в частности запросы с номерами 0—177, которые выдают на экран символ, код которого соответствует номеру запроса.

Для тех, кто захочет воспользоваться средствами, описанными в данной статье, приводится текст ПИФ. Для экономии места дается только образ памяти в восьмеричном виде. Пользователь должен будет ввести эту программу в память ЭВМ с помощью любого монитора начиная с адреса 1000.

Текст программы начальной инициализации ПИФ

```
004767 000374 012737 125636 000010 012737 001060 000034
012767 001032 000210 000137 120056 012706 001000 012737
001142 000004 012767 001120 000164 104004 000137 002000
106466 000002 010566 000002 011605 016516 177776 106216
```

```
100402 000137 121502 103404 106116 162716 103400 013607
012706 001000 014501 103005 014412 016101 014211 005002
104020 104412 104407 012706 001000 004767 000242 000000
000000 000000 000000 000000 000000 000000 000000 000000
121572 121624 122134 122074 132064 132052 122360 001120
121520 121640 121752 001120 124134 001120 122250 001120
001120 122012 122040 125326 122056 122022 121556 125556
001120 001120 000000 000000 000000 000000 000000 000000
000000 000000 000000 000000 000000 000000 000000 000000
000000 000000 000000 000000 000000 000000 000000 000000
000000 000000 000000 000000 000000 000000 000000 000000
000000 000000 000000 000000 000000 000000 000000 000000
005000 012701 001430 012705 000020 012021 077502 000207
005001 012700 001430 000767
```

В заключение отметим, что указанный подход применим и тогда, когда используется микросхема с другим языком, например с Бейсиком. Правда, в этом случае можно воспользоваться только идеей. К сожалению, ни завод-изготовитель, ни разработчики языков не предоставляют пользователям ЭВМ достаточной информации о программе, записанной в ПЗУ. Это приводит к тому, что пользователи вынуждены проводить свои исследования, подобные этому.

Приложение

Здесь приводятся все изменения, которые надо внести в текст программы ПИФ или учесть при использовании Фокала, находящегося в микросхеме К1801РЕ2-084.

1. Значения векторов прерывания 10 и 34 должны быть равны 125760 и 121300 соответственно.
2. В программе ПИФ должны быть изменены значения ячеек 1006, 1104 и 1134 на 125760, 121654 и 014313.
3. В программе ПИФ изменяется таблица TRAP-запросов, расположенная начиная с адреса 1200. Для микросхемы 084 значения этой таблицы следующие:

```
121744 121776 122306 122246 132012 132000 122532 001120
121672 122012 122124 001120 124256 121414 122622 001120
001120 122164 122212 125450 122230 122174 121730 125700
```

Литература

1. *Писаревский А. Н., Осетинский Л. Г., Осетинский М. Г.* Фокал — диалоговый язык для мини-ЭВМ. Л.: Машиностроение, 1985.
2. *Вигдорчик Г. В., Воробьев А. Ю., Праченко В. Д.* Основы программирования на Ассемблере для СМ ЭВМ. М.: Финансы и статистика, 1983.

От редакции. Рукопись уже пошла в производство, когда вышел № 5 подписной научно-популярной серии «Новое в жизни, науке, технике», серия «Вычислительная техника и ее применение», в котором опубликована эта же статья. Мы решили не отказываться от ее публикации в нашем журнале не только потому, что его тираж выше, но и потому, что затронутая тема весьма полезна для нашей более однородной и заинтересованной читательской аудитории.

О. ВОВК

Наш кабинет

Мы получили КУВТ-86. Школа у нас новая, и при ее строительстве было предусмотрено наличие кабинета вычислительной техники, т. е. была осуществлена подводка электропитания к учительскому и к каждому ученическому столу с соблюдением всех правил техники безопасности. Щиток, позволяющий осуществлять централизованную подачу (и отключение) электроэнергии, находится в лаборантской. Чтобы протянуть линию связи учительской ЭВМ с ученическими, пришлось продолбить пол и спрятать все провода, защитить их от механических повреждений. Ученические столы «пристрелили» к полу. Зашили древесно-стружечной плитой с отверстиями батареи, чтобы обезопасить работу детей, сидящих вблизи окон.

Рабочие места расположили обычным образом — три ряда ученических столов, обращенных к доске. Ведь в процессе обучения часто приходится пользоваться доской, экраном, плакатами, к тому же удобнее, когда учащиеся обращены к учителю лицом, а не боком или спиной.

Пришлось поработать над организацией рабочего места учащегося. Оно состоит из блока питания, телевизора, клавиатуры, соединительных проводов. Все это поставляется россыпью, у учащихся имеется свободный доступ ко всем узлам, разъемам. Сразу возникла мысль, что все это необходимо скомпоновать в единый блок, ограничить доступ к жизненно важным элементам. Спроектировали «блокировщик» (см. чертеж), а изготовить их помогло наше базовое предприятие — Институт автоматики и электрометрии СО АН СССР. Блокировщик состоит из двух поддонов, соединенных шарниром. Нижний поддон имеет бортики, не позволяющие учащимся лазить внутрь БК. С задней стороны — прижимная пластинка, которая позволяет зафиксировать все провода вместе. Верхний поддон служит подставкой для телевизора, который в таком положении прикрывает разъемы. К верхнему поддону прикреплено также «гнездо» для блока питания. Теперь рабочее место учащегося приобрело вид терминала (фото 1).

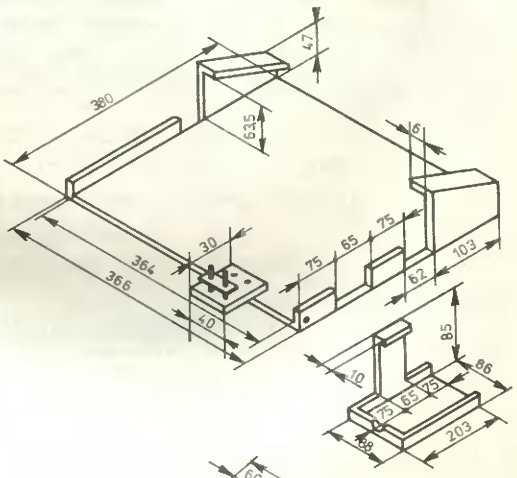
А теперь — об оформлении кабинета.

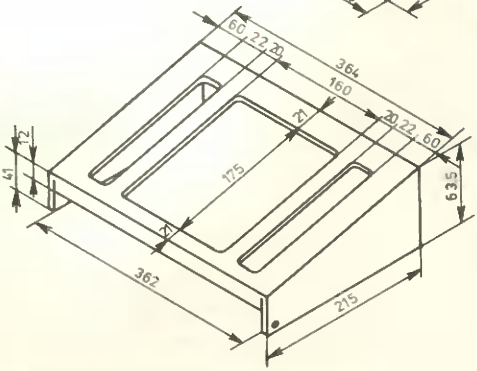
Самая главная стена — та, что у ребят перед глазами. Доска с зажимами для сменных плакатов, необходимых во время урока, экран, который можно развернуть для демонстрации слайдов или фильмов, и под потолком — длинная вереница плакатов, на которых перечислены все необходимые сведения о Фокале — операторы, директивы, функции и т. д. Удачная находка: эти плакаты мы расположили между горизонтальными полозьями в 2—3 слоя; стоит лишь сдвинуть по полозьям плакат, как за ним откроется другой.

На этой стене не стоит располагать ничего лишнего, только то, что необходимо для работы на уроке, чтобы не рассеивать внимание учащихся. А высказывания, портреты и прочую информацию можно расположить на боковых и задней стенах.

На задней стене расположены пять стендов, содержащих постоянную информацию и объединенных единым замыслом: от прошлого — к будущему.

Верхний стенд «От абака до компьютера» рассказывает о предыстории ЭВМ, о том, как люди издавна стремились усовершенствовать, облегчить процесс вычислений. Материал для этого стенда почерпнут из книги Р. С. Гутера и Ю. Л. Полунова «От абака





до компьютера», фотографии сделаны с нее же. Каждая карточка отражает очередное изобретение, говорит об авторе этого изобретения. В целом по стенду можно проследить, как человечество подошло к созданию первой ЭВМ.

Слева стенд «Как это начиналось» подробно рассказывает о создании МЭСМ и об основоположнике отечественной электронно-вычислительной техники С. А. Лебедеве. Материал для этого стенда взят из книги Л. Н. Дашевского и Е. А. Шкрабары «Как это начиналось».

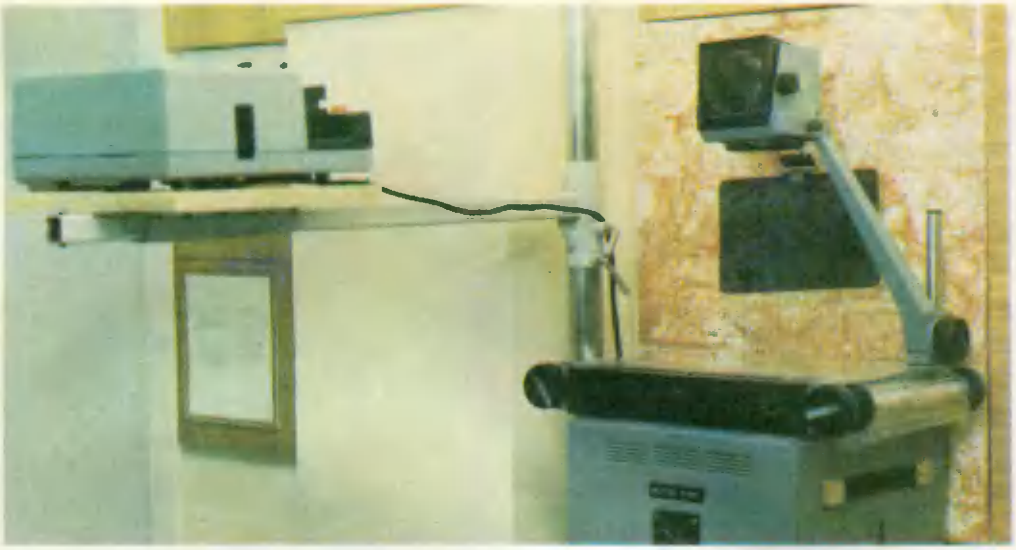
Нижний стенд «История отечественных ЭВМ» отражает все четыре поколения электронно-вычислительных машин. На каждой карточке рассказано о каких-то двух машинах: даны основные технические характеристики, когда и кем были созданы, представлены фотографии этих машин. Сведения обо всех ЭВМ можно взять в учебнике по информатике и в методическом пособии для учителей. Труднее было с иллюстрациями. Снимки приходилось отыскивать в самых различных книгах, журналах «Наука и жизнь», «Квант».

Над этими стендами расположилась галерея портретов изобретателей, ученых, которые внесли свой вклад в развитие вычислительной техники.

Следующий стенд «Области применения ЭВМ» служит хорошей иллюстрацией к уроку «Роль ЭВМ в современном обществе». На этом стенде — красочные иллюстрации, взятые из журналов как отечественных, так и зарубежных, на которых отражены самые различные сферы использования ЭВМ, кратко пояснены новые для учащихся термины: САПР, АСУ, робототехника, томография и т. д.

Логическим завершением всей композиции служит стенд «Куда пойти учиться». Здесь перечислены специальности, связанные с ЭВМ (идея позаимствована из журнала «Информатика и образование», № 1 за 1987 г.). Для учащихся, которые захотят выбрать эти специальности,— реклама учебных заведений, имеющихся у нас в Новосибирске. А внизу, в кармане из оргстекла,— проспекты многих учебных заведений с подробным описанием всех факультетов, с вариантами задач, предлагавшихся в прошлые годы на вступительных экзаменах. Десятиклассники частенько листают эти проспекты, пытаются решать задачи, желая оценить свой уровень.

Стенды, размещенные на боковой стене кабинета,— «рабочие»: на них можно разместить любую сменную информацию. Несколько карманов из оргстекла, в которых можно расположить интересный для ребят материал по различным темам. Поскольку материал мы черпаем из журналов «Квант», «Наука и жизнь», «Информатика и образо-



вание», «Техника — молодежи», «Юный техник», то и названия наших рубрик взяты оттуда же: «ЭВМ решает задачу» — о каком-то интересном применении компьютеров; «Анатомия вычислительных машин» — об устройстве ЭВМ более подробно, чем это освещается в курсе ОИВТ; «Искусство программирования» — здесь ребята могут получить какие-то сведения о других языках программирования; «В мире алгоритмов», «Веселый урок».

На стенде отведено место для творческих работ учащихся. Такими работами могут быть рефераты учеников и законченные программы, оформленные по определенным правилам: постановка задачи, описание программы, инструкция по ее использованию, распечатка текста программы.

На книжных полках расположилась небольшая библиотечка из журналов и книг научно-популярной и занимательной литературы, которую удалось собрать за два года.

На стенде «Следим за прессой» размещены вырезки из газет, отражающие достижения в области вычислительной техники, освещающие проблему компьютеризации образования, демонстрирующие внедрение вычислительной техники во все сферы нашей жизни. Этот стенд помогают обновлять сами же учащиеся.

На трех колоннах, имеющихся в нашем кабинете, нашли свое место высказывания М. В. Келдыша, Г. И. Марчука, А. П. Ершова. А вот найти их портреты даже у нас, в Новосибирском академгородке, было задачей довольно сложной.

На фото 2 можно видеть приспособление для установки технических средств обучения, названное нами «манипулятор». Конструкция его проста: на трубу, спрятанную за колонну, надевается кронштейн, по которому перемещается полка с подставкой для диа- или графопроектора. Чтобы привести в рабочее состояние ТСО, необходимо развернуть кронштейн на 90° и сдвинуть полку вдоль кронштейна так, чтобы объектив расположился напротив экрана. В конце кабинета расположен еще один такой же манипулятор, на котором разместился кинопроектор «Радуга».

Только вот остается сожалеть, что редко используются ТСО на уроках информатики — ничтожно мало поступило в школу слайдов, диафильмов. А фильмов мы еще вообще не видели. Хорошо, что у нас в школе есть вычислительная техника, которая сама о себе красноречиво говорит. А как приходится вести этот предмет учителю из глубинки, если нет возможности свои слова подкреплять демонстрацией того, о чем говоришь? Да и оформление кабинета могло бы стать для учителя менее трудоемким делом, если бы были наглядные пособия, изображающие машины, представляющие все поколения ЭВМ, портреты ученых и т. д.

Мы, учителя информатики, понимаем, что оснащение школ вычислительной техникой — дело многих лет, за год-два эту проблему не решить. А вот обеспечить все школы достаточным количеством наглядных пособий по информатике можно было бы. Для этого надо обратить на наши нужды внимание соответствующих учреждений, разрабатывающих и выпускающих наглядные пособия.

В заключение хочу выразить благодарность В. И. Яресько за помощь, оказанную им при оформлении кабинета.

В. БЕРКУЦКИЙ, И. ПОЛИВАНЫЙ, С. ЩЕГЛОВ
Павлодарский педагогический институт

Лица «Агата»

В ряде номеров ИНФО опубликованы письма пользователей ПЭВМ «Агат», содержащие резкую критику самой машины, возможностей ее использования и имеющегося программного обеспечения. Ответы на некоторые из этих вопросов и перспективы развития «Агата» изложены в обзоре [1]. Несмотря на то что обзор написан с «двойной плотностью», многие проблемы лишь обозначены, а некоторые не сформулированы. Причиной тому — их количество и многоплановость. Сталкиваясь с ними, многие пользователи начинают винить в своих бедах технику (см., например, [2, 3]). Мы хотим поделиться трехлетним опытом эксплуатации класса из 13 ПЭВМ «Агат», входящего в состав лаборатории вычислительной техники (ЛВТ) Павлодарского пединститута. Сразу оговоримся, что мы не видим альтернативы выпуску качественной техники, считаем вдвойне безнравственной поставку несовершенной техники в школы, но убеждены, что «Агаты» можно достаточно успешно эксплуатировать в школьном кабинете ВТ, а многие негативные моменты, отмеченные в [2], легко устранимы.

Начнем с техники. Вот простейшие советы, которые может реализовать любой пользователь.

Если клавиша нажимается с трудом, то пользователь невольно увеличивает усилие на нее, а это приводит к «залипанию» контакта, а то и к отрыву кнопочного стакана. Поэтому подвижную часть кнопок следует смазать смазкой «ЦИАТИМ-202». Здесь многие воскликнут: «Дожились! Компьютеры машинным маслом мажем!» Однако, проделав эту процедуру год назад, мы практически забыли, что такое неисправная кнопка. Заметим, что существует удобное программное средство для тестирования блока клавиатуры.

Чтобы одним и тем же диском пользоваться на различных машинах, необходимо, чтобы скорость вращения во всех НГМД была одинаковой. В процессе эксплуатации ее необходимо периодически контролировать и корректировать. Существует программное средство, позволяющее делать эту регулировку достаточно просто.

Диски нередко выходят из строя из-за неправильной эксплуатации. Пользователи берут диск за «окно», через которое считывается информация, что снижает работоспособность диска. В этом случае диск необходимо аккуратно, без нажима протереть через «окно» спиртом.

При неаккуратном вводе диска в НГМД может сойти со спирального желоба магнитная головка. В этом случае при загрузке ДОС происходит сбой. Необходимо снять плату «логика» и устранить дефект.

Укажем несколько простейших способов расширения возможностей «Агата».

С целью экономии дисков можно использовать обе их стороны, для чего надо снять переключатель «Защита записи» НГМД. По нашим наблюдениям, срок службы каждой стороны диска при этом не уменьшается.

В ЛВТ создана простая и очень дешевая локальная сеть из «Агатов». Это, помимо прочего, позволяет значительно уменьшить износ дисков, уменьшить зависимость от сбоев НГМД, упростить эксплуатацию кабинета ВТ. Автором этой разработки и программной поддержки является девятиклассник В. Федосеев*.

К «Агату» легко подключить цветной телевизор и использовать его на занятии как телевизор и демонстрационный монитор. Для подключения достаточно иметь всего одну микросхему К155ЛН1. Объединение «Агатов» в сеть значительно повышает эффективность использования демонстрационного монитора.

Кроме того, существует возможность создания архива на магнитной ленте.

В заключение разговора о технике уместно заметить, что мы рискнули провести «компьютерный десант» с «Агатом». Мы проехали по далеким от идеала дорогам свыше 300 км, останавливались в нескольких школах и интернате, дважды во время работы были перепады напряжений, но отказов по вине техники не было. Аналогичная поездка с КУВТ «Ямаха» представляется более рискованной, так как выход из строя НГМД практически выводит из строя весь КУВТ.

Привлекают функциональные возможности «Агата» (особенно в сравнении со всеми имеющимися в школах компьютерами). В частности, как объект изучения «Агат» (в рамках Бейсик-системы) позволяет решать широкий спектр задач, работать в интерактивном и программном режиме, предоставляет возможность выводить на экран

* Описание разработки см.: Информатика и образование. 1989. № 3.— *Примеч. ред.*

дисплея алфавитно-цифровую и графическую информацию, работать с литерными переменными, с внешней памятью под управлением ДОС, осуществлять музыкальный вывод, изучать основы программирования на языке ассемблера микропроцессора 6502 (сохраняя при этом простоту редактирования программ Бейсик-системы) и т. п. Использование в учебном процессе системы программирования «Школьница» позволяет демонстрировать достоинства структурного программирования. Не следует забывать о том, что «Агат» предоставляет каждому ученику уникальную возможность (в сравнении с КУВТ «Ямаха», КУВТ-86 и т. п.) работать с простейшими СУБД, что обусловлено наличием НГМД на каждой машине. Имеющееся прикладное программное обеспечение позволяет демонстрировать работу с электронными таблицами, редакторами текстов, музыкальными редакторами и т. п.

Как средство обучения «Агат» предоставляет разработчику широкий спектр возможностей для создания педагогических программных средств:

инверсно-мерцающий и цветной вывод алфавитно-цифровой информации; гибкие средства трехмерной статической машинной графики, позволяющие строить с помощью ограниченного набора примитивов (точка, отрезок, вектор, ломаная) достаточно сложные графические объекты (параметрические кривые, графики функций одной переменной, проекции трехмерных тел, проекции поверхностей и т. п.);

средства генерации динамических изображений (в Бейсике — операторы XDRAW, ROT, SCALE);

возможность создания новых графических примитивов (в Бейсике — DRAW);

звуковой генератор, позволяющий исполнять музыкальные произведения и имитировать человеческую речь;

простые и мощные средства ассемблера и т. п.

В обзоре [4] мы с удивлением обнаружили, что наши первые пробные педагогические программные средства составляют шестую часть ППС, имеющихся в НИИ ШОТСО АПН СССР. Это говорит о том, что разработка таких программ для «Агатов» практически не ведется. Такое же положение и на различных конференциях разработчиков ППС, где «Агатам» уделяется мизерное внимание. А ведь известно [1], что по количеству машин, имеющихся в просвещении, «Агат» уступает только «Электронике БК-0010». Таким образом, возникают большие «ножницы» между социальным заказом и сложившимся положением вещей.

В заключение — несколько предложений.

Следует помнить, что новая техника будет выпускаться теми же изготовителями. Поэтому гораздо более качественным, дешевым и более быстрым способом получения новой техники видится модернизация уже имеющейся (разумеется, речь не идет о принципиально новых разработках). В частности, модернизация ПЭВМ «Агат» (например, в направлении, указанном в [1]) должна дать школе хороший компьютер.

Необходимо количество разработок базового и прикладного программного обеспечения привести в соответствие с потребностями школы.

Хорошо бы открыть на страницах ИНФО рубрику «Полезные советы» и публиковать в ней технические «мелочи» для всех типов отечественных ПЭВМ.

Литература

1. *Кривцов А.* «Агат» в образовании: 1983—1987 //Информатика и образование. 1988. № 2.

2. *Чистопольский П.* «Как бы есть...» //Информатика и образование. 1988. № 2.

3. *Петров М.* Нужен ли в спецшколе «Агат»? // Информатика и образование. 1988. № 1.

4. *Иглицкий А., Крылова Е., Луцкий В.* Рыба ли рак? // Информатика и образование. 1987. № 1.

М. АЛЕКСЕЕВ, А. РЯБОВА
г. Миасс

«Агат», ассемблер и музыка

Наверное, многие согласятся, что одно из наиболее ярких впечатлений от первого знакомства с ПЭВМ «Агат» оставляет исполнение тест-программой мелодии Баха. А как самим запрограммировать мелодию? Ведь в штатном Бейсике «Агата» нет соответствующих операторов. Но нет худа без добра — это прекрасный повод поближе узнать машину и язык ассемблера.

За динамик в «Агате» отвечает ячейка с адресом ⌑ C030. При каждом обращении к ней на чтение или запись на динамик

подается короткий импульс напряжения, и если это делать в цикле — должен получиться звук определенной частоты. Но в цикле

FOR I=1 TO N: POKE ¤C030,0:NEXT I

мы услышим только перестук, интерпретатор работает слишком медленно. Вставленный же в Бейсик-программу фрагмент на ассемблере будет работать гораздо быстрее.

Язык ассемблера — мощный и гибкий инструмент, позволяющий реализовать буквально любой замысел. Мы же воспользуемся лишь несколькими простейшими ассемблерными командами, подобными, кстати, командам программируемого калькулятора. Они описывают операции с объявленными однобайтовыми ячейками и регистрами A, X, Y. Так, по команде LDX содержимое указанной ячейки помещается в X, а по команде STA в указанную ячейку помещается содержимое A. Для уменьшения содержимого ячейки на 1 предназначена команда DEC, то же делают с X и Y команды DEX и DEY. Понадобятся нам также команды ветвления: BNE — условный переход на указанную метку в случае ненулевого результата предыдущей операции, BEQ — то же при нулевом результате; RTS — возврат в вызывающую программу.

Входными параметрами нашей первой простейшей программы будут ПЕРИОД — условный интервал между обращениями к динамику, соответствующий периоду звуковых колебаний, и ДЛИНА — условная продолжительность звука, соответствующая числу периодов (русские буквы выбраны для наглядности, возможно, кто-то предпочтет имена типа DLINA или LENGTH). Место для размещения параметров и программы в памяти следует выбрать так, чтобы не испортить систему и свои данные. Возьмем, например, адрес ¤ 2000.


```
10 ЗВУК=¤2000
20 ПЕРИОД=ЗВУК-1
30 ДЛИНА=ЗВУК-2
70 *  ЗВУК    : REM                алг ЗВУК(цел ПЕРИОД, ДЛИНА)
80  REM                            нач
120 REM                            пока ДЛИНА>0
130 ! ДИНАМИК:LDA ¤C030    ;       нц   импульс
140 !         LDX ПЕРИОД   ;         для X от ПЕРИОД до 1 шаг -1
150 ! ЦИКЛ    :DEX         ;         нц
160 !         BNE ЦИКЛ     ;         кц
170 !         DEC ДЛИНА    ;         ДЛИНА=ДЛИНА-1
250 !         BNE ДИНАМИК ; кц
260 !         RTS          ; кон
270 !         :
300 POKE ПЕРИОД,255
310 POKE ДЛИНА,255
320 CALL ЗВУК
```

Необходимо иметь в виду различное использование имен в Бейсике и ассемблере. Например, в Бейсике ПЕРИОД — это переменная, в которую заносится число ¤ 1FFF, а в ассемблере ПЕРИОД — это ячейка, расположенная по адресу ¤ 1FFF, куда перед вызовом ЗВУКа необходимо оператором POKE занести соответствующий байт.

Комбинируя обращения к подпрограмме ЗВУК с различными входными параметрами, уже можно пытаться создавать шумовые эффекты. Однако для исполнения мелодий она неудобна, так как продолжительность исполняемых ею звуков зависит от высоты.

Для исправления этого недостатка в новой версии будем измерять продолжительность звука не периодами, длительность которых отсчитывается уменьшением содержимого регистра X на 1 в цикле от значения, записанного в ячейку ПЕРИОД, до 0, а независимыми от входных параметров интервалами, равными максимально возможному периоду (256) и отсчитываемыми уменьшением Y в цикле от 0 (256) до 0 (дело в том, что если Y=0, то команда DEY делает его равным 255).

Еще один недостаток — несохранение исходного значения параметра ДЛИНА. Это довольно неудобно, поэтому заведем рабочую ячейку ДЛ.

```
10 ЗВУК=¤2000
20 ПЕРИОД=ЗВУК-1
30 ДЛИНА=ЗВУК-2
40 ДЛ=ЗВУК-3
70 *  ЗВУК     : REM              алг ЗВУК(цел ПЕРИОД, ДЛИНА)
80  REM                           нач
100 !           LDY #0       ; Y=0   (Y=256)
110 !           LDA ДЛИНА    ; ДЛ=ДЛИНА
120 !           STA ДЛ       ; пока ДЛ>0
```

```
130 ! ДИНАМИК :LDA $C030    ; нц   импульс
140 !          LDX ПЕРИОД  ;   для X от ПЕРИОД до 1 шаг -1
150 ! ЦИКЛ    :DEY         ;   нц  Y=Y-1
160 !          BNE СЧЕТХ   ;     если Y=0 то   (Y=256)
210 !          DEC ДЛ      ;       ДЛ=ДЛ-1
220 !          BEQ ВЫХОД   ;       если ДЛ=0 то выход все
230 ! СЧЕТХ   :DEX         ;     все
240 !          BNE ЦИКЛ    ;   кц
250 !          BEQ ДИНАМИК ; кц
260 ! ВЫХОД   :RTS         ; кон
270 !          :
300 POKE ПЕРИОД,255
310 POKE ДЛИНА,255
320 CALL ЗВУК
```

В таком виде программа уже годится для исполнения быстрых мелодий, но генерировать долгие звуки она не способна. Все происходит слишком быстро даже при максимальном значении (256) однобайтового параметра ДЛИНА. Введем еще один параметр — МНОЖИТЕЛЬ — для увеличения длительности звука, определяемой параметром ДЛИНА, в заданное число раз, и рабочую ячейку МН. Программа дополнится такими строками:

```
50 МНОЖИТЕЛЬ=ЗВУК-4
60 МН=ЗВУК-5
290 POKE МНОЖИТЕЛЬ,16
80  !          LDA МНОЖИТЕЛЬ ; нач
90  !          STA МН        ; МН=МНОЖИТЕЛЬ
```


Тело цикла станет таким:

```
160 !          BNE СЧЕТХ     ;   если Y=0 то   (Y=256)
170 !          DEC МН        ;     МН=МН-1
180 !          BNE СЧЕТХ     ;     если МН=0 то
190 !          LDA МНОЖИТЕЛЬ ;       МН=МНОЖИТЕЛЬ
200 !          STA МН        ;       ДЛ=ДЛ-1
210 !          DEC ДЛ        ;       если ДЛ=0 то выход все
220 !          BEQ ВЫХОД     ;     все
230 ! СЧЕТХ   :DEX           ;   все
```

Подобным образом можно увеличить и период колебаний, чтобы понизить частоту звука до басов и даже перестука, но мы пока перестанем совершенствовать программу и займемся задачей извлечения «нефальшивых» звуков. Отношение частот соседних нот (полутон) равно корню 12-й степени из 2, значит, значения параметра ПЕРИОД должны принадлежать соответствующей геометрической прогрессии. Построим прогрессию на три октавы, взяв за начало 256 (0) — самый низкий звук.

```
300 DIM НОТА%(37) : НОТА%(1)=0
310 A=256 : P=2^(1/12)
320 FOR I=2 TO 37
330 A=A/P
340 НОТА%(I)=INT(A+0.5)
350 NEXT I
```

Выбрав одну из нот в качестве опорной и отсчитывая от нее полутона вверх и вниз, можно закодировать мелодию в виде последовательности номеров нот — элементов массива прогрессии. Для каждого звука надо задать еще и длительность, но, в связи с тем что звуки одинаковой длины зачастую соседствуют, можно сэкономить, кодируя длительность отрицательным числом и считая, что она относится ко всем последующим нотам до новой длительности: —8 — одна восьмая, —16 — одна шестнадцатая и т. п. Если ко всем номерам нот добавлять одно и то же число, получится та же мелодия, но в другой тональности — выше или ниже.

```
400 СДВИГ=5
410 READ КОД
420 IF КОД<0 THEN POKE ДЛИНА,-64/КОД : GOTO 410
430 КОД=КОД+СДВИГ
440 IF КОД>37 OR КОД<1 THEN END
450 POKE ПЕРИОД,НОТА%(КОД)
460 CALL ЗВУК
470 GOTO 410
500 REM МАЛЕНЬКАЯ ЕЛОЧКА
510 DATA -8,8,-16,5,5,-8,8,-16,5,5,8,6,5,3,-4,1,-8,
        10,-16,13,10,-8,8,-16,5,5,8,6,5,3,-4,1,256
```

Экспериментируя со сдвигом тональности, можно убедиться, что слишком высокие ноты звучат хуже, слышится фальшь, вызванная ошибками округления до целого. Для сравнения распечатайте члены прогрессии с округлением и без него. Работая с параметром МНОЖИТЕЛЬ, можно изменять темп исполнения — andante, allegro и т. п.; вставляя между всеми звуками небольшие паузы, можно получить staccato. Вообще кодирование паузы (видимо, нулем) и ее отработку тоже обязательно надо предусмотреть. И собрать мелодии в библиотеку, и сделать каталог мелодий, и музыкальный редактор, и еще, и еще... Развивать эту тему можно бесконечно.

Конечно, в некоторых версиях Бейсика есть специальные операторы SOUND и PLAY, а в Рапире — стандартные процедуры ЗВУК и НОТА, облегчающие программирование музыки. Но, овладев ассемблером, вы можете попытаться сымитировать звук выстрела или удара, шум мотора и даже синтезировать речь. Желаем успехов!

Литература

Техническое описание ПЭВМ «Агат». Книга 2. Фг.3.032.002.ТО1—3. ППП «Школьница». Описания Рапиры и Отладчика.

Морер У. Язык ассемблера для персонального компьютера Эпл. М.: Мир, 1987.

А. ГРИШАЕВ
с. Малиновка

На Луну на МКШ-2

В обстановке «повальной» компьютеризации сельские школы чувствуют себя за бортом корабля, плывущего по волнам информатики. Не случайно многие сельские учителя испытывают раздражение от нового предмета; а что говорить об учениках?

Понятно, что лучший способ приобщить детей к ЭВМ — это игра. Но нам пока не к чему приобщать — мы располагаем лишь простейшими вычислительными средствами, например ЭКВМ МКШ-2. В печати редки сообщения о разработке игр для простейших микрокалькуляторов; между тем можно играть и на них.

На такую мысль нас натолкнул анализ игровой программы «Посадка на Луну». Возможностей МКШ-2 вполне хватает, чтобы произвести необходимые вычисления; кроме калькулятора для проведения игры используется таблица исполнения, заполняемая по результатам вычислений.

Итак, на Луну падает ракета с собственной массой M=500 кг и запасом топлива M1 (в начальный момент M1=250 кг). Расстояние до Луны равно H (в начальный момент H=15 км), скорость падения V (сначала — 1 км/с). Включается тормозной двигатель и в течение времени, равного Tc, расходует L кг/с топлива; раскаленные газы вылетают из сопла двигателя со скоростью 3 км/с, создавая ускорение (замедляющее падение ракеты) A задача игрока — так подобрать расход топлива и длительность включений тормозного двигателя, чтобы ракета мягко села на Луну, т. е. V=0 при H=0.

Для решения этой задачи нужно знать еще ускорение свободного падения в окрестностях Луны; поскольку оно меняется незначительно, можно задать его таблицей:

H	15 000—10 000	10 000—8000	8000—4000	4000—0
G	1,58—1,59	1,59—1,6	1,6—1,61	1,61—1,62

Задав величины расхода топлива L и времени торможения T, вычисляем ускорение A (программа для МКШ-2):

3000 [×] L [÷] [(] 500 [+] M1 [)]
[—] G [=] A [X—П]

Оно должно быть меньше 49 — иначе чрезмерная перегрузка приводит к гибели экипажа.

Если ускорение в пределах нормы, вычисляем скорость:

A [×] T [—] [+] [=] V′ [=] V

где V′ — предыдущее значение скорости (такое же обозначение применяем и в дальнейшем).

Вычисляем текущее значение высоты:

H′ [—] [(] V [×] T [)]
[+] [(] [П—X] [×] T [F] [x^2]
[÷] 2 [)] [=] H

Остаток топлива:

L |×| T |/—/| |+| M1′ |=| M1

Результаты заносятся в таблицу и становятся исходными данными для следующего этапа полета.

Пример исполнения программы:

L	T	G	A	V	H	M1
4,0	12	1,58	14,42	783,7	4038	202
11	2	1,6	45,4	692,9	2561,4	180
11	2	1,61	46,9	599	1269	158
11	2	1,61	48,54	501,9	169,36	136
10,5	1	1,62	47,9	454	—309,7	125

Как видите, на этот раз наша ракета образовала на Луне новый кратер. Но решение задачи существует, и даже с экономией топлива для последующего взлета.

Конечно, применение ЭКВМ с несколькими регистрами памяти существенно упрощает вычисления, но вопросы оптимизации подсчетов каждый может решить самостоятельно.

Когда цель достигнута и ракета благополучно прилунилась, поставьте другую задачу: наибольшая экономия топлива, наименьшее число шагов, наименьшее время посадки, достижение наибольшей высоты на остатках топлива и т. д.

Преимущества игры на ЭКВМ относительны, но они есть: поняв, что сделанный шаг неверен и ведет к катастрофе, можно не возвращаться к началу, как это заставил бы сделать компьютер, а переиграть 1—2 шага.

Счастливой посадки!

М. СТЕПАНОВА

Компьютер и самочувствие школьника

Из множества вопросов, связанных с компьютеризацией образования, пожалуй, один из самых важных — самочувствие школьника, работающего на ЭВМ.

Интенсификация и формализация учебной деятельности, значительное зрительное напряжение, увеличение гиподинамии, специфические условия окружающей среды в дисплейных классах и др. могут оказать неблагоприятное влияние на здоровье школьников.

До последнего времени медицинская наука не располагала данными о влиянии занятий с компьютером на детский организм. Хотя исследования в этом направлении начаты совсем недавно, но уже получены первые результаты (1, 2, 3). Основная задача этих исследований — установить влияние компьютерной деятельности на растущий организм и определить оптимальные условия для этой деятельности, предупреждающие переутомление школьников. Степень утомления школьников зависит от целого ряда как объективных, так и субъективных факторов (рис. 1).

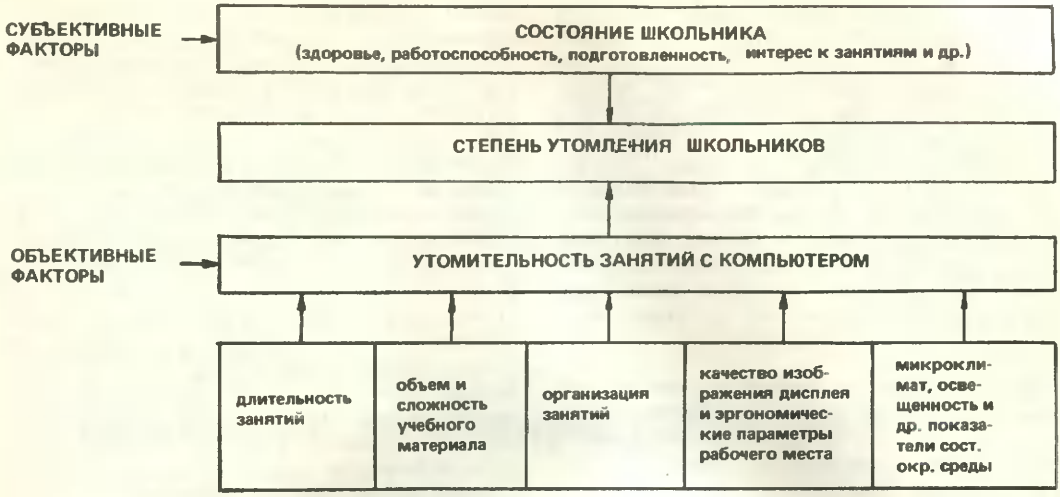
Проводимые исследования посвящены главным образом изучению влияния на функциональное состояние организма школьников объективных факторов, которые, как правило, поддаются количественной оценке и гигиеническому регламентированию.

Прежде всего это относится к длительности и режиму занятий, микроклиматическим параметрам, освещенности, различного рода излучениям, организации рабочего места, качеству изображения дисплея и др. Однако при прочих равных условиях степень утомления школьника весьма сильно зависит от субъективных факторов, работоспособности, интереса и подготовленности к занятиям.

Субъективные факторы весьма переменчивы и трудноизмеримы. Многие из них, например такие, как подготовленность, интерес к занятиям, эмоциональное состояние ученика, преимущественно зависят от педагогического воздействия.

В нашем исследовании проводилось специальное анкетирование учащихся VII—X классов (481 человек) нескольких московских школ, направленное на изучение таких важных субъективных факторов, как интерес школьников к занятиям на ЭВМ и их самочувствие после этих занятий. Старшеклассники занимались с компьютерами на уроках информатики, а учащиеся VII—VIII классов — на уроках математики. Одна часть школьников (VII—X классы — 222 человека) работали на ЭВМ ДВК-2, а другая (IX—X классы — 259 человек) на ЭВМ БК-0010, у которых в качестве дисплея использовались портативные телевизоры «Электроника-404».

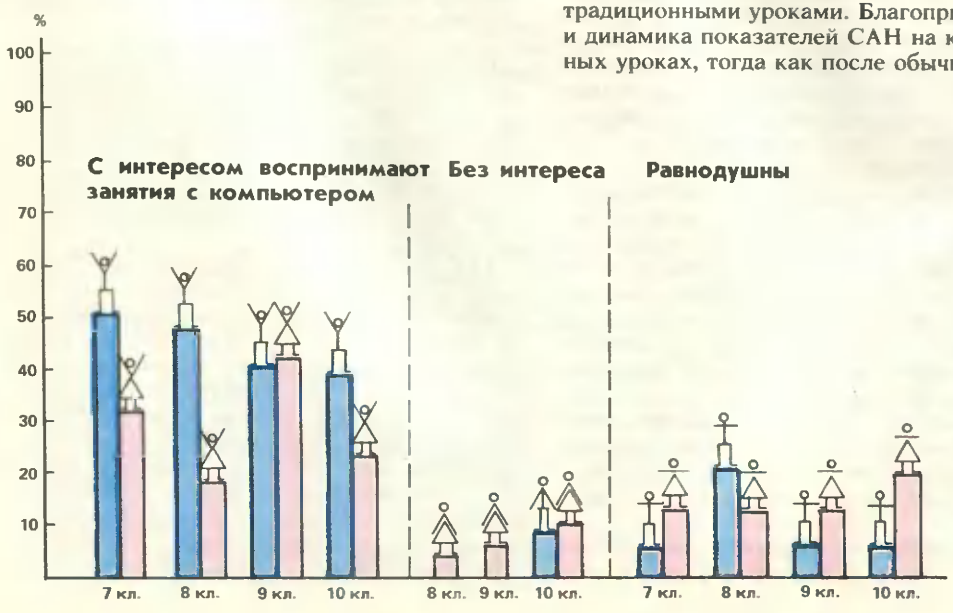
Результаты анкетирования выявили, что у большинства школьников (63,5 % старшеклассников и свыше 70 % учащихся VII—VIII классов) занятия с компьютером проходили в среднем один раз в неделю. Они



были соответственно 2—3 раза в неделю у 33,9 % и 26,7 % школьников. И лишь немногие (2,6 % и 3,1 % школьников) занимались на ЭВМ более трех раз в неделю. У основной части учеников компьютерная деятельность ограничивалась уроками (67,2 и 78,4 %), остальные, и таких несколько больше в VII—VIII классах, занимались с компьютером и во внеурочное время.

Какова же средняя продолжительность этих занятий? Почти 75 % школьников VII—VIII классов и 66 % старшеклассников указали, что продолжительность их занятий за компьютером не превышала 35 мин, а у половины из них длительность этих занятий была в пределах 25 мин. Часть школьников, причем главным образом старшеклассники, засиживались за экраном дисплея значительно дольше, а у некоторых это время измерялось даже часами.

Известно, что интерес к выполняемой работе является мощным фактором, противодействующим наступлению утомления. Наше анкетирование показало, что интерес к компьютерной деятельности у школьников разного возраста и пола неодинаков (рис. 2). Среди семиклассников подавляющее большинство (82,5 %) отметили, что занятия с компьютером им интересны. Подтверждением этому служат и результаты, полученные с помощью теста САН (самочувствие, активность, настроение). Так, у школьников седьмых классов перед уроками математики с использованием компьютеров самочувствие и настроение оказались выше, чем перед традиционными уроками. Благоприятна была и динамика показателей САН на компьютерных уроках, тогда как после обычных уроков



эти показатели, как правило, снижались. Чем старше школьники, тем меньше проявляли они интерес к занятиям с компьютером. Уже среди восьмиклассников оказались ребята, которым эти занятия неинтересны, т. е. у них уже сформировалось негативное отношение к компьютеру. Почти в 2 раза (31,0 %) по сравнению с семиклассниками возросло количество детей с индифферентным отношением к этим занятиям. Больше всего учащихся с негативным отношением к компьютерной деятельности оказалось среди старшеклассников — почти 10,0 %, втрое больше, чем среди восьмиклассников.

Меньший интерес старшеклассников к работе за дисплеем, полагаем, связан с тем, что у многих из них уже сформировались те или иные познавательные интересы, которые главным образом определяются представлениями о будущей профессии, и компьютерная деятельность в эти представления не вписывается. В этих рассуждениях мы опираемся на данные Г. Ц. Молонова (4) о том, что на формирование познавательных интересов более чем у половины старшеклассников оказывает влияние та или иная профессиональная ориентация. Не последнюю роль в формировании отношения к компьютерной деятельности играет еще и то, что методическое оснащение и качество преподавания основ информатики, а также сама компьютерная техника не отвечают современным требованиям.

Другой интересный факт, который был выявлен в результате анкетирования: мальчики, независимо от возраста, больше, чем девочки, интересуются занятиями на ЭВМ. Одна из причин этого кроется, вероятно, в том, что интересы к учебным предметам у мальчиков и девочек существенно различаются. Девочки чаще отдают предпочтение предметам гуманитарного цикла, «нетехническим». Анализ полученных анкет показал, что у девочек независимо от возраста и типа ЭВМ, на которой они работали, в 1,5 раза чаще возникали трудности в работе. Кроме того, педагоги отмечают и более выраженное у них чувство страха перед сложной техникой, особенно на начальном этапе ее освоения. Среди всех школьников, которые указывали на затруднения, возникающие при общении с ЭВМ, более 60 % — девочки. А если учесть еще и тех, кто уклончиво ответил на вопрос о затруднениях в работе на компьютере, то число таких учениц станет еще больше. Мы наблюдали и таких девочек, которые сознательно избегали общения с ЭВМ, ограничиваясь теоретическими занятиями.

Эти данные свидетельствуют о том, что, во-первых, обучение компьютерной грамоте нужно начинать не в старших классах, а раньше, используя возможности ЭВМ для преподавания общеобразовательных предметов и в кружковой работе. Во-вторых, встает вопрос о необходимости определенной психолого-педагогической работы по формированию интереса к занятиям на ЭВМ у девочек. Можно предположить, что эти вопросы связаны между собой.

Как же оценивают сами ребята свое самочувствие после работы на компьютере? Более 22 % учащихся VII—VIII классов отметили, что работа за дисплеем вызывает утомление, а в IX—X классах количество таких учащихся достигало 31,2 %, причем больше всего (44,1 %) их было среди десятиклассников. Это во многом обусловлено тем, что старшеклассники дольше работают за компьютером, уровень же их интереса к этим занятиям, как указывалось ранее, ниже. Больше жалоб на утомление отмечалось у школьников, для которых компьютерная деятельность неинтересна,— 61,1 %, тогда как среди интересующихся этими занятиями жалоб меньше — в 46,5 % случаев. Эти данные еще раз подтверждают, что работа с увлечением субъективно менее утомительна. Зрительное напряжение при работе за дисплеем обусловлено прежде всего неудовлетворительным качеством изображения (особенно если видеотерминалом служит бытовой телевизор), в поле зрения находятся поверхности различной яркости, и в связи с этим глаза испытывают постоянную переадаптацию. Это, конечно, не может не беспокоить, ведь ухудшение зрения у детей нарастает от 3—4 % в I классе до 25 % в старших и каждый пятый выпускник школы из-за близорукости имеет ограничения в выборе профессии.

Необходимо отметить, что учащиеся, которые работали на БК-0010, чаще жаловались на утомление, особенно зрительное. Если среди работавших на ДВК-2 жалобы на усталость предъявляли 21,2 % учащихся VII—VIII классов и 26,2 % старшеклассников, то среди тех, кто работал на БК-0010, их было почти 40 %.

Как показал наш анализ, наиболее уязвимы в этом отношении учащиеся с теми или иными отклонениями со стороны зрения. Из 38 % старшеклассников, которые жаловались на утомление глаз после работы на ЭВМ, 21,6 % имели нарушения рефракции. При частом и длительном общении с такой техникой их зрение станет еще хуже. Несомненно, что работа на компьютерах плохого качества приводит не только к ухудшению самочувствия, но и к потере интереса школьников к компьютерной деятельности. Это подтверждает требования гигиенистов

(5) о том, что школы и другие детские учреждения недопустимо оборудовать электронно-вычислительной техникой, не отвечающей современным требованиям.

Литература

1. *Глушкова Е. К.* с соавт. Гигиенические проблемы применения компьютеров в учебном процессе средней общеобразовательной школы // Гигиена и санитария. 1988. № 6. С. 19—22.

2. *Гельтищева Е., Селихова Г.* Режим работы за дисплеем // Информатика и образование. 1987. № 1. С. 82—92.

3. Гигиенические проблемы компьютеризации общеобразовательной школы: Сб. научн. трудов. М., 1988. 189 с.

4. *Молонов Г. Ц.* Интересы школьников к учебным предметам. Улан-Удэ, 1975. 101 с.

5. *Маслов А., Таиров Р., Труш В.* Физиолого-гигиенические аспекты использования персональных ЭВМ в учебном процессе // Информатика и образование. 1987. № 4. С. 79—80.

Г. НУРМУХАМЕДОВ,
доктор техн. наук, НИИ СиМО АПН СССР

Знакомство с элементной базой ЭВМ

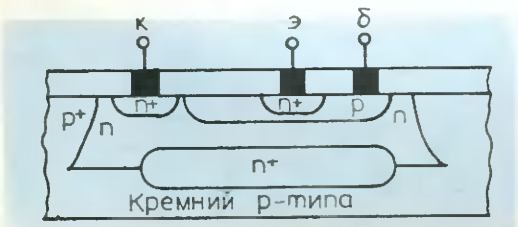
В одном из разделов программы курса ОИВТ ставится задача познакомить учащихся с основами устройства ЭВМ на примере микрокомпьютера. В данной статье даются основные сведения об элементной базе микроэлектроники.

1. Понятие об элементной базе ЭВМ. Два типа транзисторов. Полупроводниковые интегральные схемы

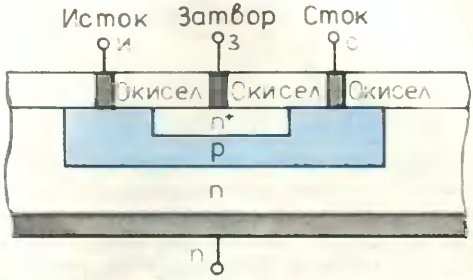
Вычислительные устройства, в которых кодирование и обработка информации осуществляются в двоичной системе счисления, называются цифровыми устройствами. Они состоят из множества электронных элементов, которые электрическим воздействием легко перевести в одно из двух устойчивых состояний. Элементы цифровых устройств предназначены для запоминания информации, ее арифметической и логической обработки, формирования и усиления сигналов управления, преобразования и отображения выходной и входной информации и т. д.

Основой большинства элементов современных ЭВМ является транзистор — полупроводниковый прибор, способный преобразовывать электрические сигналы. Существует два типа транзисторов: биполярный с двумя взаимодействующими электронно-дырочными переходами (рис. 1) и униполярный, или полевой (рис. 2). В полевом транзисторе управляющий электрод — затвор — изолирован от тела полупроводника слоем диэлектрика, обычно двуокиси кремния, вот почему этот прибор называют еще МОП- или МДП-транзистором, подчеркивая его структуру: металл — окисел (диэлектрик) — полупроводник. Из сравнения рис. 1 и 2 видно, что изготовить МДП-транзистор проще, чем биполярный, так как во втором случае на поверхности подложки — полупроводника *n*-типа — достаточно лишь сформировать две небольшие области полупроводника *p*-типа и покрыть всю поверхность слоем окисной пленки, в то время как в первом случае нужно провести два процесса формирования микрообластей с разной проводимостью.

1



2



Интегральная схема (ИС) функционально является законченным логическим, запоминающим или каким-либо вспомогательным элементом цифрового устройства. Конструктивно ИС выполняется на монокристаллической пластинке кремния размером в несколько квадратных миллиметров путем формирования с помощью специальной технологии отдельных микрокомпонентов в соответствии с функциональной схемой. Из-за того что конструкция полевого транзистора проще, чем биполярного, миниатюризацию структурных элементов ИС, использующих МДП-технологию, осуществить легче. При одинаковой функциональной сложности

МДП ИС занимают площадь на кристалле в несколько раз меньше, чем биполярные. Кроме того, из-за более простой технологии изготовления МДП-приборов на их основе возможно делать ИС с большей функциональной сложностью, чем на биполярных полупроводниках. Сейчас по МДП-технологии разработаны большие (число компонентов более 1000) и сверхбольшие (число компонентов более 10 000) интегральные схемы (БИС и СБИС). Однако МДП ИС имеют и существенные недостатки. Главный из них — сравнительно низкое быстродействие. По этому параметру биполярные ИС превосходят МДП в 10 и более раз, однако их потребляемая энергия существенно больше, чем МДП ИС. Таким образом, каждый тип ИС имеет свои достоинства и недостатки, которые и определяют в конечном счете их место в электронной аппаратуре.

2. Представление о технологии изготовления полупроводниковых интегральных схем

Первый образец транзистора был создан в 1948 г., а затем понадобилось долгих 11 лет поисков технологических решений, чтобы научиться делать простейшие интегральные схемы на биполярных полупроводниках. В результате родилась так называемая планарная технология, которая к настоящему времени настолько усовершенствована, что по праву считается одной из наиболее изящных технологий, разработанных человечеством.

Технология изготовления ИС включает в себя совокупность механических, физических и химических способов обработки различных материалов (полупроводников, диэлектриков, металлов) с применением трех классов физико-химических процессов: удаления, нанесения и перераспределения вещества.

В планарной технологии используются процессы окисления, фотолитографии, диффузии, эпитаксиального наращивания пленок, химического и ионно-плазменного травления и ряд других вспомогательных операций. В зависимости от структуры полупроводниковых приборов и схемотехнических особенностей ИС эти операции проводятся в различной последовательности.

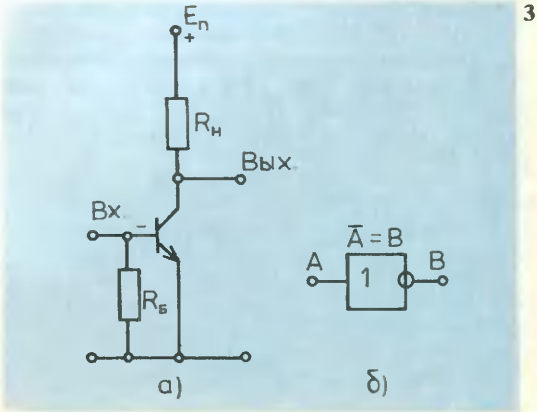
В настоящее время достигнуто воспроизведение схемных элементов размеров в 1—2 мкм, а в отдельных случаях даже до 0,3—0,5 мкм. Фактически это уже физический предел малости рисунка ИС. Столь малые размеры деталей полупроводниковых приборов позволяют на одном кристалле размером 6×6 мм^2 разместить до 100—300 тыс. отдельных компонентов схемы, иными словами, создавать очень сложные функциональные схемы: накопитель для внутренней памяти ЭВМ, центральный процессор и т. д.

По планарной технологии ИС изготовляют группами. На одной пластине кремния диаметром 100 мм одновременно формируются десятки и сотни отдельных ИС. Сейчас для размещения схемных деталей ИС средней степени интеграции отводится площадь 2,5×2,5 мм^2, для БИС — площадь 6×6 мм^2, а для СБИС — до 10×10 мм^2. После завершения процесса формирования ИС пластина кремния разрезается на отдельные кристаллы, которые затем монтируются в корпуса и проходят тщательный контроль на пригодность схемы к работе.

3. Структура и принцип работы базовых электронных элементов

Все многообразие устройств ЭВМ базируется на сравнительно ограниченном наборе типовых электронных элементов. Поэтому принцип действия даже сверхсложного компьютера легко понять, если предварительно разобраться в структуре и принципе работы базовых электронных элементов, к которым относятся инвертор (ключ), вентиль, триггер.

Инвертор. На рис. 3, а представлена схема электронного ключа на биполярном транзисторе, реализующая логическую функцию «НЕ» (отрицание), а на рис. 3, б — его условное обозначение. При подаче на вход схемы сигнала низкого уровня (логического «0») транзистор будет заперт, т. е. ток через него проходить не будет, и на выходе будет сигнал высокого уровня (напряжение источника питания $E_п$, логическая «1»). Если же на вход схемы подать сигнал высокого уровня (логическую «1»), то транзистор «откроется», начнет пропускать электрический ток. На его выходе за счет падения напряжения на сопротивлении нагрузки $R_н$ установится напряжение низкого уровня (логический

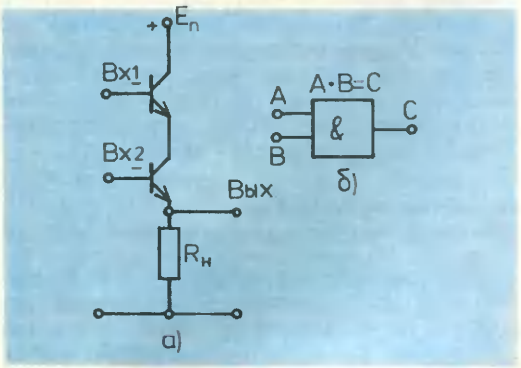


3

«0»). Таким образом, схема преобразует (инвертирует) сигналы одного уровня в другой, тем самым выполняя логическую функцию «НЕ».

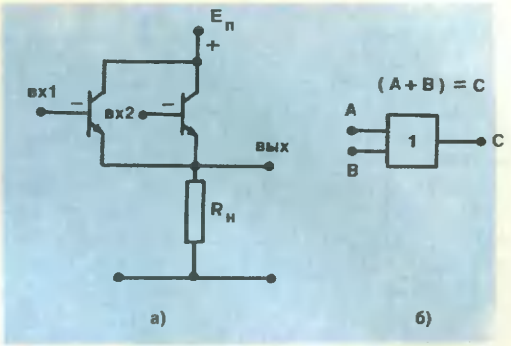
Вентиль. На рис. 4, а изображена схема вентиля на биполярных транзисторах, реализующего логическую функцию «И», а на рис. 4, б — его условное обозначение. Функция «И» — логическое умножение, ее результат С равен единице, когда оба аргумента, и А, и В, равны единице.

4



Если на входы $Вх_1$ и $Вх_2$ поданы сигналы низкого уровня (логические «0»), то оба транзистора закрыты, ток через них не проходит, выходное напряжение на R_n близко к 0. Пусть на один из входов подано напряжение высокого уровня (логическая «1»). Тогда соответствующий транзистор откроется, однако другой останется закрытым, и ток через транзисторы и сопротивление нагрузки R_n по-прежнему не будет проходить. Следовательно, при подаче напряжения высокого уровня лишь на один из транзисторов схема не переключается и на выходе остается напряжение низкого уровня. И лишь при одновременной подаче на входы сигналов высокого уровня (логических «1») на выходе мы также получим сигнал высокого уровня: открытые транзисторы практически не оказывают сопротивление току, все напряжение падает на сопротивлении нагрузки, потенциал вывода Вых становится высоким.

На рис. 5, а приведена схема вентиля на биполярных тразисторах, реализующего логическую функцию «ИЛИ», а на рис. 5, б



дано его условное обозначение. Функция «ИЛИ» — логическое сложение, ее результат С равен единице, если хотя бы один из аргументов равен единице.

Здесь транзисторы включены параллельно друг другу. Если оба закрыты, то их общее сопротивление велико и на выходе будет сигнал низкого уровня (логический «0»). Достаточно подать сигнал высокого уровня (логическую «1») на один из транзисторов, как схема начнет пропускать ток и на сопротивлении нагрузки установится также сигнал высокого уровня (логическая «1»).

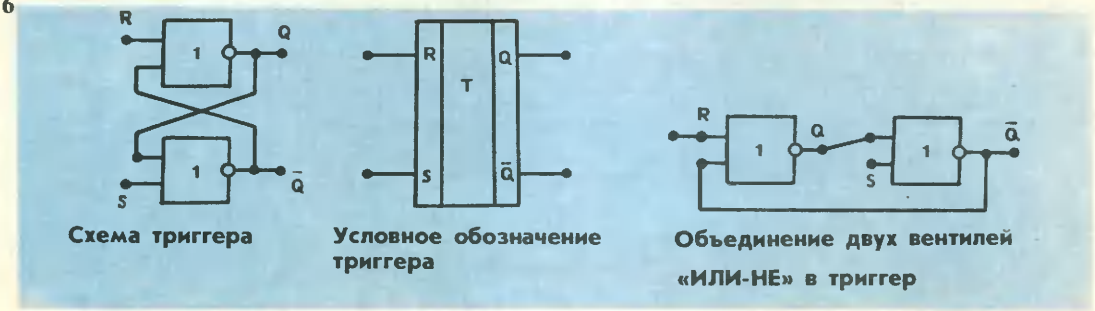
Доказано, что любая логическая функция может быть разложена на комбинацию функций «НЕ», «И-НЕ» и «ИЛИ-НЕ», так что из инвертора и соответствующих вентилей можно построить электронную логическую систему, выполняющую любое запланированное действие. Для получения вентилей «И-НЕ» и «ИЛИ-НЕ» из «И» и «ИЛИ» достаточно перенести сопротивление нагрузки R_n из эмиттерной цепи в коллекторную (как в схеме инвертора).

Триггер. Триггером называется электронное устройство с двумя устойчивыми состояниями, одно из которых характеризуется высоким (логическая «1»), а второе низким (логический «0») уровнем выходного сигнала. Триггер состоит из двух вентилей.

На рис. 6 показан способ объединения в триггер двух вентилей «ИЛИ-НЕ» (точно так же для этой цели используются и вентили «И-НЕ»).

Рассмотрим работу этой схемы. Пусть в начальный момент времени входы R, S

6



и выход Q имеют низкий логический уровень. Для переключения триггера в состояние Q=1 необходимо на вход S подать «1». На входе соответствующего вентиля будут действовать входные логические сигналы «0» (с выхода Q) и «1» (со входа S). На его выходе возникает инвертированная «1», т. е. «0». Следовательно, через некоторое время Δt, в течение которого входной сигнал S=1 достигнет выхода вентиля, состояние выхода Q изменится с «1» на «0». Теперь на входы второго вентиля будет действовать новая пара сигналов: «0» на вход R и «0» с выхода $\bar{Q}$. Следовательно, еще через Δt на выходе этого вентиля возникнет инвертированный сигнал «0», т. е. «1». Таким образом, через время $2\Delta t$ после подачи входного сигнала S=1 на выходе Q триггера логический «0» изменится на логическую «1». Следующее переключение триггера произойдет, если на вход R подать сигнал высокого уровня, и т. д. Триггер может работать бесперебойно лишь с периодом, не меньшим $4\Delta t$. В современных транзисторных вентилях Δt составляет единицы наносекунд (10^{-9} с), поэтому быстродействие электронных элементов вычислительных устройств очень большое, достигающее десятков и сотен миллионов переключений в секунду.

Регистр. Из триггеров (они бывают и других типов, отличных от рассмотренного) строятся многие элементы ЭВМ, например регистры. Они предназначены для приема, временного хранения и передачи информации в двоичном коде. Каждый триггер регистра используется для ввода, хранения и вывода одного разряда двоичного числа. Регистр, предназначенный для хранения информации, называют накопительным. Существуют также сдвигающие регистры, в которых двоичную информацию можно перемещать поразрядно влево и вправо, а также счетные регистры, предназначенные для преобразования десятичных чисел в двоичные и обратно.

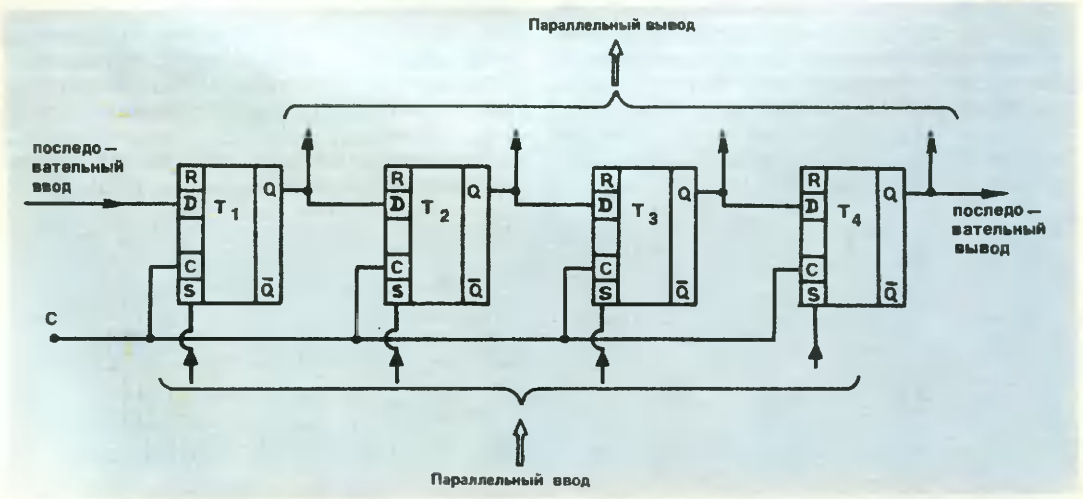
На рис. 7 приведена схема регистра, способного хранить 4-значное двоичное число. Информацию можно записывать в него и считывать как в параллельном коде (т. е. подавая или снимая все разряды числа одновременно по четырем шинам), так и в последовательном (подавая или снимая разряды числа по очереди). Этот регистр собран на D-триггерах, отличающихся от рассмотренного (который называют RS-триггером) тем, что информацию подают на вход D, и записывается она только в том случае, когда на вход C подается разрешающий сигнал — логическая «1».

4. Состав и назначение основных микросхем электронной техники

Наиболее распространенной и универсальной по функциональному назначению является серия микросхем К155. Изменяя набор базовых элементов и структуру их соединений, на кристаллике микросхемы формируют регистр, сумматор, арифметико-логическое устройство (АЛУ), дешифратор, мультиплексор, демультиплексор и др.

О назначении регистра было упомянуто выше. Назначение сумматора ясно из названия: он предназначен для арифметического сложения чисел в двоичном коде. Чтобы реализовать операцию сложения одноразрядного двоичного числа, необходимо соединить определенным образом 9 вентилей «И-НЕ», из них 5 предназначены для формирования кода суммы, а 4 — для кода переноса из данного разряда в следующий. Для обработки многоразрядных двоичных

7



чисел одноразрядные сумматоры последовательно подсоединяются друг к другу в необходимом количестве. При этом, однако, увеличивается время вычислений.

Для устранения этого недостатка разработаны схемы АЛУ, в которых осуществляется одновременный перенос в каждом разряде. В этих схемах реализованы функции не только арифметических, но и логических операций, поэтому они сложнее и насыщеннее электронными компонентами, чем схемы простых сумматоров. Так, в микросхеме 4-разрядного АЛУ используется 5 инверторов, 33 вентиля «И», 2 вентиля «И-НЕ», 8 вентилей «ИЛИ» и 13 вентилей «ИЛИ-НЕ».

Еще одна микросхема серии К155 — дешифратор, устройство, использующееся для преобразования двоичного кода на входе в определенную комбинацию сигналов на выходе. Например, дешифратор на 3 входа и 8 выходов состоит из 8 вентилей «И» и осуществляет преобразование одной из восьми возможных комбинаций двоичных сигналов на входе в единичный сигнал на одном из 8 выходов. Дешифратор широко используется в качестве промежуточного устройства между процессором и памятью и выполняет роль «диспетчера», определяющего нужную ячейку памяти для записи или чтения информации.

Наконец, мультиплексор — устройство, имеющее несколько входов и один выход. Он обычно применяется в качестве коммутатора сигналов для преобразования параллельного кода в последовательный. Для обратного преобразования используется демультиплексор.

Рассмотренные варианты микросхем серии К155 относятся к логическим устройствам средней степени интеграции. Кроме этого, ряд основных блоков и узлов выполнен на микросхемах большой и сверхбольшой степени интеграции. Это прежде всего устройства памяти и микропроцессоры. Современные запоминающие устройства для оперативной и постоянной памяти ЭВМ формируются на кристаллах размером до 6×6 мм^2 со степенью интеграции до 64К бит на одной микросхеме КР565РУ5. Еще более сложна схемная реализация универсального 16-разрядного микропроцессора серии К1801ВМ1. Структура, принцип действия и характеристики этих микросхем подробно описаны в учебно-наглядном пособии для учащихся X класса (Нурмухамедов Г., Кузнецов А. Структура и принципы работы ЭВМ: Учебно-наглядное пособие для 10 класса. 13 таблиц и объяснительный текст для учителя. М.: Просвещение, 1987).

Вниманию читателей

Государственный комитет СССР по народному образованию, Союз учителей, Вычислительный центр СО АН СССР (г. Красноярск), Красноярский государственный университет, центр НТТМ «Спец» предполагают в апреле 1990 г. провести в г. Красноярске Всесоюзную конференцию учителей информатики.

Основной целью конференции является создание секции информатики Союза учителей и фонда методических разработок и программного обеспечения.

В рамках конференции состоится детский компьютерный фестиваль.

Конференция продлится неделю. В первой половине дня предполагается проведение лекций, семинаров, дискуссий, обмен опытом, открытых уроков, показ программного обеспечения. Во второй половине дня желающие могут стать участниками конкурсов детского фестиваля, оказать помощь в их организации, проведении, стать членами жюри.

Хотелось бы услышать предложения учителей по проведению конференции, созданию секции информатики Союза учителей.

Оргкомитет будет ждать ваших писем по адресу: 660051, г. Красноярск, а/я 11533. Оргкомитет конференции учителей информатики.

ВЛАДЕЛЬЦЫ КУВТ-86!

В нашем кооперативе ***«Альтернатива»*** создан пакет учебных программ для поддержки школьного курса «Основы информатики и вычислительной техники».

Пакет является реальной альтернативой всем ранее созданным программным системам для КУВТ-86 (система «Рига», 36-часовые курсы для КУВТ-86-Фокал и КУВТ-86-Бейсик и т. д.).

В него входит более 100 программ и файлов, в том числе:

клавиатурные тренажеры;

система подготовки текстов;

программы-исполнители;

система управления базами данных;

электронная таблица;

Е-практикум;

музыкальный редактор;

программа «Электронный диск», обеспечивающая свободный доступ каждого БК-0010 к дисководу и ОЗУ ДВК-2М, а также быструю и параллельную пересылку программ по сети в КУВТ-86 любой модификации;

программа «Меню», обеспечивающая простой и наглядный выбор пользователем БК-0010 программы, с которой он хочет работать, и загрузку этой программы. Управление автоматически передается в «Меню» при нажатии клавиши «СТОП»;

интерпретатор Бейсика, пересылаемый по сети (для КУВТ-86-Фокал);

система программирования на ассемблере для БК-0010;

пакет игровых программ для БК-0010;

обучающие программы по отдельным учебным предметам;

инструкции ко всем входящим в пакет программам;

инструкции к КУВТ-86 и операционной системе;

методические рекомендации по использованию пакета в учебном процессе.

Все программы пакета составлены в машинных кодах или на ассемблере БК-0010, поэтому безразлично, Фокал или Бейсик загружен в ПЗУ БК-0010. Это сводит практически на нет различия между разными модификациями КУВТ-86 и резко облегчает методическое обеспечение учебного процесса.

Все программы пересылаются по сети и могут осуществлять ввод и вывод на дисковод ДВК-2М или на электронный диск в его ОЗУ.

Большинство программ предусматривают возможность подключения к БК-0010 джойстика.

Почему мы взялись за создание этого пакета?

Мы считаем, что КУВТ-86 не является достижением в области вычислительной техники, но, создав для них должное программное обеспечение, можно, а следовательно, нужно привести их в соответствие с требованиями программы средних учебных заведений.

Мы считаем, что использование КУВТ-86 только для изучения программирования на Бейсике или Фокале — дилетантство, допустимое разве что на этапе первоначального освоения.

Мы считаем, что учебные программные системы для КУВТ-86 следует разрабатывать только в кодах БК-0010 или на ассемблере. Это — единственный способ уложиться в жесткие ограничения по быстродействию и памяти. Попытки создать серьезные программы на иных языках (Фокал, Бейсик, Паскаль, Т-язык, Си) обречены на неудачу — программы будут технически примитивны и методически скудны. И не следует принимать эти неудачи за неизбежное следствие технического несовершенства КУВТ-86. Они — результат неверного подхода.

Мы считаем, что на службу народному образованию следует поставить все лучшие достижения в области программирования, поэтому в некоторых случаях

адаптировали программы, созданные не нами. Права их авторов при этом не ущемлялись.

Приобретение нашего пакета позволит вам значительно повысить технический и методический уровень использования КУВТ-86.

Приобретение пакета подразумевает заключение договора, строго ограничивающего круг пользователей, для которых приобретается пакет. Цена зависит от широты круга будущих пользователей (школа, город, район и т. д.).

За дополнительную плату пакет комплектуется джойстиками.

В перспективе — перенесение пакета на КУВТ УКНЦ.

Поставка — на наших дискетах (8 шт.).

Для приобретения пакета обращайтесь по адресу: 656099, Алтайский край, г. Барнаул, пр. Социалистический, 60, каб. 15.

ПРЕДЛАГАЕТ

ТЕРМИНАЛ

Персональный компьютер — не только объект изучения, но и средство обучения. С его помощью намного глубже, быстрее и полнее можно постигать не только математические, физические или химические знания, но и историю, биологию, географию,— одним словом, любую из точных или гуманитарных наук.

Но для этого нужна специальная программа. **Такая программа есть:**

ДЖИНН.

Ее создал известный советский программист **А. Б. Борковский,** автор «Англо-русского словаря по программированию и информатике». Естественно, она знает любое из содержащихся в этом словаре слов и выражений и готова мгновенно выдать на экран перевод любого из них. Кроме того, ДЖИНН готов объяснить любой специальный термин, встречающийся в любых программах, растолковать любую команду операционной системы MS DOS.

Но главное, его можно — и мы готовы сделать это по Вашему заданию — научить чему угодно. Например, любому учебному курсу или же порядку проведения любых лабораторных работ. И в этом случае он сумеет прокомментировать любое непонятное место любого текста, дать определение любого термина, установить связь между различными понятиями. Ибо **в основе программы ДЖИНН лежит идея гипертекста — одна из самых современных и самых плодотворных в программировании.**

Размещенная на четырех дискетах программа ДЖИНН стоит **всего 297 рублей.**

Программа ДЖИНН — фирменный продукт московского НТК «Терминал». Брошюра с подробным описанием и бланки заказов высылаются бесплатно.

Справки по телефону: 427-07-88.

Адрес для переписки: 121360, Москва, НТК «Терминал».

Л. ШТЕРНБЕРГ
канд. физ.-мат. наук
Куйбышевский авиационный институт

Уроки с программируемыми микрокалькуляторами

Табличные величины

В соответствии с приведенным ранее вариантом поурочного планирования табличные величины даются в конце года (уроки 29—30). У школьников этот материал вызывает трудности, связанные с понятиями «индекс» и «значение». Чтобы их преодолеть, можно перед табличными величинами рассмотреть работу с литерными величинами: там проблема снимается тем, что индекс — число, а элемент величины — литера.

В любом варианте для обработки табличных и литерных величин лучше вначале использовать цикл раз: алгоритмы получаются короче и понятнее. Кроме того, именно так придется программировать на ПМК. Сравним тела алгоритма «сумма таблицы»:

```
сум:=0; i:=1
пока i⩽N
нц
  сум:=сум+a [i]; i:=i+1
кц
```

```
сум:=0; i:=1
цикл N раз
  сум:=сум+a [i]; i:=i+1
кц
```

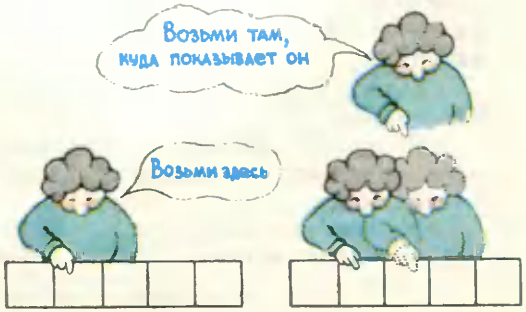
Здесь естественным образом вводится цикл с параметром, в котором заголовок цикла вбирает в себя и начальную установку параметра, и его изменение, и проверку на окончание цикла.

Введя табличные величины, желательно рассмотреть те задачи, которые связаны с их обработкой: они рассеяны по разделам «Решение задач из курса математики» и «...физики»: вычисление многочлена, поиск в таблице заданного и минимального элемента, упорядочение, метод наименьших квадратов и т. д.

Проработав хотя бы часть этих алгоритмов, можно приступать к их реализации на ПМК. Эта тема достаточно сложна из-за того, что на ПМК нет средств, соответствующих индексированию (обращению к элементу таблицы) в «чистом виде». Поэтому приходится использовать преобразование алгоритма с введением дополнительных средств в алгоритмический язык (А-язык). Кроме того, необходимое средство ПМК — косвенная адресация — само по себе достаточно сложно для восприятия. Поэтому эту тему, видимо, не следует считать обязательной и добиваться ее усвоения всеми.

Из-за малой памяти ПМК практически невозможно реализовать работу с двумерными таблицами, поэтому рассмотрим работу с одномерными.

Для того чтобы команда, работающая в цикле, при каждом новом исполнении выбирала (или засылала) число из (или в) другого регистра, она должна изменяться



Продолжение. Начало см.: Информатика и образование. 1989. № 2, 3, 4.

в процессе исполнения программы невозможно. Поэтому номер регистра, с которым работает команда, нельзя задавать в самой команде (заметим, что вторая цифра кодов команд «П—х» и «х—П» — это и есть номер регистра), а надо задавать в таких местах, которые доступны для изменений в процессе выполнения программы,— ими являются адресуемые регистры и регистры стека. Так, учитель должен подвести к понятию косвенной адресации (на ПМК косвенная адресация осуществляется только через адресуемые регистры). Понятие косвенной адресации можно объяснить на примере, не связанном с ПМК (см. рис. 1). Далее надо напомнить, что с косвенной адресацией мы имели дело при занесении программы в память ПМК: команда попадает туда, куда указывает счетчик адреса, а затем счетчик увеличивается (только здесь он увеличивается не до, а после засылки). Для иллюстрации работы команд косвенной адресации можно провести следующий эксперимент (класс при этом работает в «командном режиме»): засылаем в регистры числа, равные номеру регистра (0 — в P0, 1 — в P1, ... 13 — в PD), и выполняем команды (см. табл. 1).

Таблица 1

Команда	Индикатор	Комментарий
К П—х 4	5.	Вызов произошел из P5; только там есть число «5»
К П—х 4	6.	Теперь та же команда вызвала число из P6
К П—х 4	7.	Та же команда вызвала число из P7
П—х 4	000 000 07.	А в P4 — уже число 7 (в необычном виде)

Аналогично проводим демонстрацию косвенной записи, косвенной работы с регистрами, работающими с автоуменьшением и без автоизменений. Полностью во всех деталях этот эксперимент описан в [1].

Итак, ясно, что если команду «К П—х М» или «К х—П М» заставить работать в цикле, то она может обрабатывать последовательность регистров, хранящих разные элементы таблицы.

Но в А-языке для обработки таблиц используется индексация, а на ПМК — косвенная адресация. Следовательно, перед переводом алгоритма с А-языка на язык ПМК его нужно преобразовать так, чтобы заменить индексацию на косвенную адресацию. В учебнике понятия косвенной адресации в А-языке нет. Перед учителем возникают два варианта: а) ввести в А-язык дополнительное понятие (особенности конкретной ВТ всегда накладывают отпечаток на преподавание), б) не вводить дополнительное понятие и выходить из сложившейся ситуации объяснениями на интуитивном уровне, как это сделано в учебнике, где неявно используются сразу две не введенные в явном виде конструкции: косвенная адресация и цикл до (с проверкой условия в конце). Сомневающимся можно предложить попробовать оба варианта и лично убедиться, что проще ввести дополнительную конструкцию, чем объяснять, как обходиться без нее. Поэтому здесь рассмотрен только вариант б).

Разница между индексом и адресом заключается в том, что адрес — это *абсолютный* номер регистра, отсчитываемый от начала массива регистров, а индекс — *относительный* номер, отсчитываемый от начала той группы регистров, которую занимает данная таблица, причем счет ведется, возможно, не с 1, а с нижней границы диапазона индекса, указанного в описании таблицы. Например, для таблицы ...таб A [*k*:*m*] адрес элемента A [*i*] равен $m+(i-k+1)$, где m — номер первого из регистров, отведенного для хранения таблицы. Введем в А-язык две новые конструкции:

адрес *переменная*

которая не нуждается в пояснениях, и косвенную адресацию, соответственно простую, с автоувеличением и с автоуменьшением, записываемую:

|*k*| |⁺*k*| |⁻*k*|

и означающую обработку переменной, адрес которой записан в *k*. Для переменных, используемых в косвенной адресации, лучше ввести и специальные типы косв, косв— и косв+, так как эти типы следует учитывать при распределении регистров.

При обработке табличных величин обычно используют цикл для. Для ПМК его придется преобразовать к циклу раз, что делается по строго формальным правилам:

```
для k от a до b шаг c нц
...
кц
```

→

```
k:=a;
цикл (b — a)/c+1 раз
...
k:=k+c
кц
```

Далее вводится переменная с описателем косв, которая заменяет индекс, все обращения к элементам таблицы заменяются на косвенную адресацию, а начальное присва-

ивание индексу заменяется на присваивание косвенной переменной адреса соответствующего элемента массива.

Преобразуем, например, алгоритм «сумма таблицы»:

```
алг  сумма таблицы (вещ таб A [1:6], вещ сум)
   арг A; рез сум
нач цел i
   сум:=0;
   для i от 1 до 6 нц
      сум:=сум+A [i]
кц
кон
```

преобразование цикла

```
сум:=0; i:=1
цикл 6 раз
   сум:=сум+A [i]; i:=i+1
кц
```

Теперь переходим к косвенной адресации:

```
нач цел i
   сум:=0; i:=1
   цикл 6 раз
   сум:=сум+A [i]; i:=i+1
   кц
```

→

```
нач косв k
   сум:=0; k:=адрес A [1]
   цикл 6 раз
      сум:=сум+ [k]; k:=k+1
   кц
```

Как видим, преобразование выглядит достаточно просто. Правила программирования алгоритмов с этими конструкциями таковы:

1. Распределение памяти начинается с выделения регистров для переменных с описателями **косв**, **косв—** и **косв+**, так как для них подходят не все регистры.

2. Конструкция **адрес** программируется командами набора соответствующего числа.

3. Конструкции [k], [^{+}k] и [^{-}k] программируются командами «К П—х ...» или «К х—П ...» с номером отведенного переменной k регистра.

Применим эти правила к рассмотренному алгоритму (см. табл. 2).

Заметим, что при одновременной обработке двух и более таблиц в А-языке можно обойтись одним индексом, а на ПМК нужно иметь свою косвенную переменную для каждой таблицы.

Если требуется обработать все элементы таблицы в порядке их расположения, то удобно применять косвенную адресацию с автоувеличением или автоуменьшением. Процесс преобразования алгоритма таков (подчеркнуты меняющиеся элементы):

```
косв k
сум:=0; k=адрес A [1]
цикл 6 раз
   сум:=сум+ [k]; k:=k+1
кц
```

Таблица 2

Адрес	Команда	А-язык	Комментарий
счетчик — Р0 k — Р7 А[1:6] — Р8—РD сум — Р1		алг сумма косв k (вещ...А... сум)	Распределение памяти начинается с отведения регистров счетчику цикла и косвенной переменной, далее отводим место прочим переменным
00 01 02 03	0 х—П 1 8 х—П 7	сум:=0 k:= адрес А[1]	Адресом А[1] является 8; набираем «8» в РХ и засылаем адрес в k
04 05	6 х—П 0	цикл 6 раз	Количество повторений цикла засылаем в счетчик (обычный способ реализации цикла раз)
06 07 08 09	П—х 1 К П—х 7 + х—П 1	сум:= сум+[k]	Работа с косвенной переменной реализуется командой косвенной адресации (в данном случае чтения)
10 11 12 13	П—х 7 1 + х—П 7	k:= k+1	Увеличиваем адрес, содержащийся в k
14 15 16	F L0 06 С/П	кц кон	Завершаем цикл стандартным для цикла раз образом

Таблица 3

Адрес	Команда	А-язык	Комментарий
			Команды 00...01 без изменений
02 03	7 x — П4	*k*:= адрес А [0]	Элемента А [0] в таблице нет, но если бы он был, то занимал бы Р7
04 05	6 x — П 0	цикл 6 раз	Заголовок цикла программируется так же
06 07 08 09	П — x 1 К П — x 4 + x — П 1	сум:= сум+ [^{+}k]	А здесь косвенная адресация помимо чтения нужного числа еще и увеличивает значение *k*
10 ...	F L 0 ...	кц ...	Далее все без изменений

```
косв k
сум:=0; k:=адрес А [0]
цикл 6 раз
  k:=k+1; сум:=сум+ [k]
кц
```

```
косв— k
сум:=0; k:=адрес А [0]
цикл 6 раз
  сум:=сум+ [+k]
кц
```

Теперь для *k* можно взять Р4, и, оставив остальное распределение памяти без изменений, получим программу (см. табл. 3).

Здесь имеется широкое поле для оптимизации программы: хранение суммы непосредственно в РХ, размещение таблицы в обратном порядке (А [6] — в Р8, А [5] — в Р9) и ее обработка с применением адресации с автоуменьшением и т. д. Все это описано в [1] на с. 75—79, и эти варианты можно рассмотреть в кружках или классах с математическим уклоном, для урока в обычном классе можно рекомендовать только простейший вариант (можно и не обсуждать косвенную адресацию с автоизменением, упомянув только о ее существовании и не рекомендуя использовать для нее Р0 ...Р6).

По этой теме для проработки на ПМК можно взять множество имеющихся в учебнике задач по обработке таблиц.

Сочетания программных конструкций

В наших примерах были рассмотрены программы, содержащие только по одной конструкции нужного нам вида: одно ветвление или один цикл. Именно на таких программах и следует пояснять работу конструкции. После введения каждой конструкции в отдельности можно перейти к задачам, где требуется применить две или более конструкции: два цикла, цикл и ветвление и т. д. Конструкции сочетаются только двумя способами: *следованием* и *вложением*. Следование конструкции при программировании проблем не вызывает, а при программировании вложений учащиеся допускают много ошибок. Учитель должен сформулировать правило программирования сочетаний разных конструкций и следить за его выполнением, тем более что формулируется оно просто: «Каждая конструкция программируется по своим правилам, независимо от того, что следует за ней или за чем следует она, а также, что вложено в нее или во что вложена она».

Проследим реализацию этого правила на примере алгоритма подсчета числа отрицательных элементов таблицы, где алгоритм содержит цикл, в который вложено ветвление (без ветви иначе) (см. табл. 4).

Здесь возможны попытки учеников «оптимизировать» программу, рассуждая так: если элемент неотрицателен, то подсчитывать его не надо, а значит, надо просто перейти к обработке следующего элемента, т. е. с адреса 08 перейти на адрес 06. В итоге в ячейке 08 появляется адрес 06 — и в программу внесена ошибка: так как не выполнена команда «F L0», то будет неправильным число повторений цикла (вплоть до зацикливания). Чтобы ошибок не было, нужно приучать учеников к строго формальному переводу алгоритмов на язык ПМК: адрес заполняется при обработке все (и не ранее) очередным значением счетчика адреса.

Возможности оптимизации программ за счет команд переходов и использования других особенностей ПМК действительно

Таблица 4

Адрес	Команда	А-язык	
счетчик — P0 A[1:8] — P5—PC кол — P1 k — P4		алг число отриц. (таб A[1:8] цел кол) нач косв+k	
00 01	0 x—П 1	кол:=0	
02 03	4 x—П 4	k:= адрес A[0]	
04 05	8 x—П 0	цикл 8 раз	Элементы программы, необходимые для цикла типа раз
06 07 08	К П—x 4 F x<0 13	если [^{+}k] <0 то	Элементы программы, необходимые для программирования ветвления
09 10 11 12	П—x 1 1 + x—П 1	кол:= кол+1	
13 14	 FL0 06	все кц	
15	С/П	кон	

нередко появляются, но они не должны быть темой уроков, а лишь предметом рассмотрения на кружках.

Работа со стеком

Этот материал (если учитель решит познакомить с ним учащихся) должен быть дан на 12—14-м уроке, однако рассмотрение было до сих пор отложено, так как он является дополнительным: приемы работы со стеком — это особенность ПМК данного семейства, они выйдут из употребления вместе с уходом ПМК с «арены» школьной информатики, в то время как все остальное (за исключением разве что косвенной адресации) приложимо без изменений к переводу алгоритмов, например, на Бейсик.

Вместе с тем этот материал математически красив, легко доступен и хорошо воспринимается учащимися как удобное средство выполнения цепочечных вычислений.

Использование стека связано с математической конструкцией, называемой *польская инверсная запись* (ПОЛИЗ), изобретенной в 1921 г. польским математиком Я. Лукасевичем. Обычно в учебниках и справочниках по ПМК мало говорится о ПОЛИЗе, без объяснения, что это такое; в результате на уроках появляются «задачи»: в каком порядке вычислять сложное выражение, чтобы было меньше вспомогательных действий (запоминаний/вызовов чисел, обменов между регистрами и т. д.). Эти задачи не имеют права на существование, ибо ПОЛИЗ как раз и позволяет, не задумываясь, получать оптимальную последовательность команд.

Идея ПОЛИЗа заключается в том, что знак операции записывается после своих

операндов, причем сам операнд может быть выражением, записанным в ПОЛИЗе:

	ПОЛИЗ
$a-b$	$a\ b\ -$
$sin\ (x)$	$x\ sin$
$a\cdot(b-c)$	$a\ b\ c\ -\ \times$
$a\cdot b-c$	$a\ b\ \times c-$

Как показывают два последних примера, в ПОЛИЗе нет скобок и нет старшинства операций: они не нужны — операции выполняются в порядке их записи, что и есть основное преимущество ПОЛИЗа.

Для вычисления записанного в ПОЛИЗе выражения нужен стек, который можно представлять (эту аналогию учитель должен дать обязательно) как магазин стрелкового автомата: коробочка с пружиной, в которую только с одной стороны можно втолкнуть патрон (в нашем случае патрон будет не с пулей, а с числом). Вынуть его можно тоже только с одной стороны. Правила вычисления выражения на ПОЛИЗе:

1. Выражение просматривается один раз слева направо.
2. Числа (значения переменных или константы) помещаются в стек.
3. Если встретилась операция, то из стека выбирается столько чисел, сколько нужно для операции (одно или два), над ними выполняется операция, результат помещается назад в стек.
4. В конце в стеке остается только окончательный результат.

Но для вычисления выражения, записанного в обычной форме, нет необходимости в предварительном преобразовании в ПОЛИЗ. Рассуждая определенным образом, можно получить оптимальную программу вычисления выражения за один его просмотр.

1. Выражение записывается в одну строку (без многоэтажных формул) и просматривается один раз слева направо.
2. Константы и значения переменных заносятся в стек.
3. Операция выполняется немедленно после того, как в стеке оказались все ее операнды.

Рассмотрим реализацию этих правил на примере выражения

$$\frac{sin\ (3\cdot a-\frac{b}{2})}{c-d}.$$

Таблица 5

Элемент	Рассуждения	Действие	Стек X	Стек Y	Стек Z	Стек T
sin	Пока невыполнимо: отложим					
3	Константу заносим в стек	3	3			
x	Пока невыполнимо: отложим					
a	Значение переменной — в стек и можно выполнить умножение	П — x A ×	*a* $3\cdot a$	3		
—	Пока невыполнимо: отложим					
b	Значение переменной — в стек	П — x B	*b*	$3\cdot a$		
/	Пока невыполнимо: отложим					
2	Значение константы — в стек	2	2	*b*	$3\cdot a$	
	и можно выполнить «/»: делим	÷	*в*/2	$3\cdot a$		
	А теперь можно вычитать, и можно «синусовать»	— F sin	$(3\cdot a-$ $-b/2)$ *sin* (...)			
c	Значение переменной — в стек	П — x C	*c*	*sin* (...)		
—	Операция пока откладывается					
d	Значение переменной — в стек	П — x D	*d*	*c*	*sin* (...)	
	И остается выполнить отложенные операции	— ÷	$c-d$ рез-т	*sin* (...)		

Считаем, что a находится в РА, b — в РВ, c — в РС, d — PD. Обратите внимание, что *за тем, где что находится в стеке, следить не надо: все автоматически оказывается на своих местах.* Поэтому закройте пока листом бумаги часть табл. 5, на которой изображен стек (на уроке это тоже следует сделать). Запись выражения в одну строку имеет вид: $sin\ (3 \cdot a - b/2)/(c - d)$ (знаменатель пришлось взять в скобки). Читаем его слева направо, скобки читать не будем (они только учитываются для определения порядка действий).

Теперь можно открыть правую часть таблицы и посмотреть, как «качаются» операнды в стеке (в примере для лучшей иллюстрации взяты неперестановочные операции: деление, вычитание).

ПОЛИЗ для нашего выражения имеет вид: $3\ a \times\ b\ 2\ /\ -\ sin\ c\ d\ -\ /$

Если сравнить его с полученной программой, то становится ясно, почему говорят, что ПМК работает на идее ПОЛИЗа.

Если приходится набирать две константы подряд, то их надо разделять командой «B↑», а команды «FO» и «←→» при классической работе со стеком вообще не используются. Регистр T потребуется в более сложных выражениях, а ситуацию, где стека не хватает, и придумать-то непросто. Технические подробности можно посмотреть в [1] на с. 11—18 или [2].

Продолжение следует.

Литература

1. *Штернберг Л. Ф.* Программирование на микрокалькуляторе. М.: Просвещение, 1988.
2. *Штернберг Л. Ф.* Зачем микрокалькулятору стек // Квант, 1986. № 4.

Е. ЛИНЕЦКИЙ

Городской компьютерный центр: опыт, проблемы, перспективы

О Свердловском городском компьютерном центре (ГКЦ) уже упоминалось в [1]. Цель настоящей статьи — более подробно рассказать о ГКЦ, описать конкретное применение предложенного в [1] подхода к обучению, обосновать оптимальность организации обучения в ГКЦ.

Наш ГКЦ объединяет под одной крышей специализированный программистский учебно-производственный комбинат, преподавание школьной информатики (для учащихся школ, не оснащенных вычислительной техникой), школу юных программистов и методический центр для учителей информатики. Мы стараемся не забывать и о сельских школьниках. С 1987 г. организуются недельные практические занятия по информатике для старшеклассников из разных районов области. Теоретические вопросы они изучают дома по тому же учебнику, что и городские школьники (см. [1]). Программа полного практического курса — 48 ч. Таким «вахтовым» способом в 1987/88 учебном году около тысячи сельских учащихся смогли поработать на ЭВМ.

В ГКЦ установлено 48 ЭВМ «Роботрон 1715» (операционная система SCP — роботроновская версия хорошо известной CP/M) — это четыре компьютерных класса. Кроме того, есть два теоретических класса. Учащиеся группы информатики и юные программисты занимаются 2 ч в неделю, а учащиеся УПК приходят к нам раз в неделю на целый учебный день — 6 ч, из которых первые два занимаются теорией, а остальное время (с необходимыми перерывами) работают с ЭВМ. Каждый год в ГКЦ почти 400 школьников обучаются в УПК, более 1700 — в группах информатики и около 70 посещают занятия школы юных программистов.

Наша идеология курса ОИВТ изложена в [1], функции методического центра очевидны, поэтому речь пойдет в основном о профильном обучении в группах УПК и о школе юных программистов (ШЮП), хотя все формы обучения в ГКЦ взаимосвязаны.

ШЮП возникла при Свердловском пединституте в 1982 г., а в 1986 г. «переехала» в ГКЦ. Опыт работы в ШЮП во многом помог нам почти на самом старте школьной компьютеризации предложить вполне сформировавшийся подход к обучению информатике. В обучении юных программистов с самого начала стали выделяться два направления. С одной стороны, нужно было организовать увлекательный компьютерный ликбез для массы любознательных детей (в ШЮП принимаются ученики IV—VIII классов), а с другой — придумать занятие для тех юных

программистов, у которых ликбез занял месяц (а не год, как это в среднем бывает) и которые хотели бы двигаться дальше. Развитие первого направления привело к серии популярных публикаций в областной молодежной газете, послуживших затем основой для первых глав нашего учебника информатики. Существует и специальная методическая разработка для ШЮП [2].

В процессе развития второго направления, который сейчас продолжается не только в рамках ШЮП, но и в группах программистов УПК, родилось понимание того, чему и как следует обучать школьников, проявивших особые способности к программированию.

Центральной идеей и целью преподавания как ОИВТ, так и программирования мы считаем обучение умению решать задачи с помощью ЭВМ. Под задачами понимаются не обычные школьные математические, физические или химические задачи, предполагающие применение более или менее шаблонных методов. Принципиальное отличие в том, что стандартные школьные задачи решаются в готовом, «поставленном» виде, мы же включаем в процесс решения все этапы движения от вопроса, заданного в реальной «жизненной» ситуации, к ответу, интерпретированному снова в «жизненном», а не математическом смысле.

Итак, решением задач с помощью ЭВМ в ГКЦ занимаются все дети. Поэтому, с одной стороны, следует сформировать общее для всех ядро знаний и умений (как видите, слово «навыки» мы опускаем, зато знания делим на собственно знания и «понимания», но об этом чуть позже), необходимых для решения с помощью ЭВМ вообще любых задач, а с другой стороны, предложить каждой категории обучаемых определенный специальный класс задач. Стало быть, понимание того, чему учить ту или иную категорию детей, есть понимание того, на какой класс задач мы собираемся их ориентировать. Этим определяются и содержание учебного плана, и методика, и, насколько возможно, организация процесса обучения.

Прежде чем описать специфику задач, решаемых учащимися различных форм обучения в ГКЦ, расскажем о базовом ядре знаний, умений и «пониманий». Знания и умения — категории традиционные. «Навыки» мы исключили, поскольку вычислительная техника развивается слишком быстро для того, чтобы какие-то навыки работы с ней стоило специально формировать. Под «пониманием» подразумевается нечто, отвечающее не на вопросы «что?» и «как?», а в основном на вопросы «почему?». Пример: что такое процессор и как он работает — знание, но для конкретного процессора бессмысленно ставить вопрос: «Почему он так работает?» Этот вопрос относится на самом деле ко всей истории развития вычислительной техники. Можно сказать, что понимание — это знание о нем самом, в частности о том, как оно возникло.

Вот, схематически, какую структуру базового ядра знаний, умений и пониманий мы стремимся сформировать у каждого нашего ученика:

Знания. Алгоритм и исполнитель алгоритмов; допустимые действия и достижимые цели; основные алгоритмические конструкции, их универсальность; математическая модель; этапы решения задач с помощью ЭВМ; последовательная детализация как метод решения сложных задач; связь организации данных и организации действий в алгоритмах; принципы программного управления фон Неймана; автоматическая обработка информации; роль ЭВМ в жизни и производстве; перспективы развития ЭВМ.

Умения. Работа с учебными или профессиональными текстовыми редактором, электронной таблицей, информационно-поисковой системой; реализация с помощью этих средств, других прикладных программ или языка программирования всех этапов решения задачи такого уровня сложности, при котором без последовательной детализации трудно обойтись.

Понимания. Путь развития вычислительной техники; информация — важнейший национальный ресурс; моделирование — основа решения задач; возможность построить модель — критерий понимания задачи; смоделировать можно далеко не все; ЭВМ не может творить; ЭВМ помогает решать задачи и в то же время все лучше поддерживает диалог с человеком.

Теперь пора рассказать, какие задачи мы считаем специфическими для обучения в ШЮП и УПК (на информатике задачи в силу общеобразовательности курса ОИВТ носят в основном чисто учебный характер).

Что касается ШЮП, то ее программа преследует две цели: обслужить любознательных детей и начать готовить будущих программистов для УПК из тех, у кого возник серьезный интерес к предмету. Дело в том, что сейчас многие способные учащиеся УПК чувствуют в себе силы создавать (и действительно создают) довольно сложные программы, не обладая настоящей программистской культурой, что не может не сказываться на качестве. А время, когда они могли бы приобрести необходимую культуру (IV—VIII классы), часто оказывается упущенным. В IX классе, когда они вовсю программируют, заинтересовать их

изучением основ уже весьма трудно. Кстати, эта проблема некоторыми авторами сублимируется в форме ненависти к Бейсику (впрочем, не обязательно только к Бейсику; Дейкстра, например, не любит еще и Фортран).

На каких же задачах лучше всего воспитывается программистская культура? Ответ на этот вопрос получен нами в результате довольно долгой работы в ШЮП и подтверждается двухлетней работой в УПК. Программистская культура лучше всего воспитывается в создании тренажеров, тестов, имитаторов, иллюстраторов, короче — в создании программ, объединяемых названием «педагогические программные средства» (ППС).

Какое счастливое совпадение! Ведь хорошие ППС — это как раз то, чего так не хватает школе. (Заметим кстати, что хорошее ППС на плохой машине не сделать, а от плохого учителя самое лучшее ППС не защитит.) Однако в перечислении типов ППС нет ли серьезного пробела? А что же обучающие программы? Разве они — не ППС? Здесь требуется

Лирическое отступление 1.
Что такое обучающая программа?

Деление ППС на тренажеры, тесты, имитаторы исполнителей и обучающие программы довольно традиционно. Эти термины все понимают примерно одинаково. А вот в толковании термина «обучающая программа», напротив, наблюдается удивительная пестрота.

Существует странное представление, что ЭВМ (т. е. программа ЭВМ) способна обучать, иначе говоря, «давать знания». Правда, носители этого представления достаточно отрицательно относятся к «обучающим» программам, предъявляющим ученику на экране подозрительно знакомый текст из учебника да еще заставляющим читать этот текст не в том порядке и темпе, который удобен обучаемому, а в том, который почему-либо кажется автору «обучающей» программы оптимальным. Но почему-то считается, что если снабдить такой «электронный учебник» обратной связью, расставить в нужных местах затейливые контрольные вопросы и задания (чтобы следить за процессом усвоения материала и учитывать индивидуальность обучаемого), активно пользоваться при этом идеологией баз знаний и методами искусственного интеллекта (чтобы сообщать не просто знания, а систему знаний и заодно накапливать «педагогический опыт» для применения в будущем), пригласить в качестве консультанта хорошего психолога, то в конце концов получится нечто действительно обучающее.

На самом деле получится, как говорил академик В. М. Глушков, реактивный двигатель, установленный на телегу, хотя и телега, и реактивный двигатель — вещи полезные. Вопрос не в том, нужно ли в ППС применять передовую информационную технологию, а в том, как это делать. Когда мы смотрим, скажем, на электронный микроскоп, мы становимся умнее не в большей степени, чем когда наблюдаем скромно лежащие в футляре очки. Но если мы посмотрим не на очки, а сквозь них, а потом не на микроскоп, а в его окуляр, мы почувствуем разницу между этими двумя средствами познания мира без всякого психолога.

Платон считал, что обучение — это не «наполнение сосуда», а «зажжение факела». Обучающая программа — это не то, что пытается запихнуть некие знания в голову обучаемого, перехитрив его лень и безразличие, а то, что помогает ему самому эти знания открыть. ЭВМ может сделать это благодаря самой ценной своей способности — способности к моделированию, имитации предметного мира. Последовательное проведение нашего принципа «модель — программа — вычислительный эксперимент» приводит к представлению об обучающей программе как о гибкой динамической модели некоторой предметной области с широкими возможностями для изменения как параметров модели, так и соответствующих «законов природы».

Итак, способные ученики и в ШЮП, и в группах УПК занимаются одним и тем же — пишут ППС. Правда, некоторые учащиеся УПК, кроме этого, выполняют заказы предприятий (сделаны, например, программа автоматизации кассовых расчетов в цеховой столовой, пакет бухгалтерского учета для строительного треста, завершается выполнение особенно крупного заказа — АСУ районного узла связи), а также пишут подробные и понятные инструкции по работе с системными программами, документация к которым зачастую либо отсутствует, либо характеризуется полным пренебрежением интересами пользователя.

Но главное различие между ШЮП и УПК в том, что для юных программистов создание ППС — это все-таки больше игра, чем работа, и их программы приносят пользу в основном только им самим; программисты же УПК должны обеспечивать ППС не только себя (т. е. УПК), но и информатику и ШЮП. Именно должны, потому что система «одни пишут, другие используют» пока еще не может обеспечить достаточно быстрого и качественного решения проблемы ППС. Программы, написанные школьниками под руководством преподавателей-практиков, иногда оказываются более профессиональными, часто более адекватными педаго-

гическому процессу и всегда более дешевыми, чем программы, сделанные специалистами. Это еще одно подтверждение необходимости обучать школьников так, чтобы они могли под руководством преподавателя создавать настоящие качественные ППС.

А что делать с менее способными учениками? Ведь обучать всех хорошему программированию невозможно, да и не нужно. Как уже было сказано, юные программисты, удовлетворившие свое любопытство, уходят сами.

Найдено решение и для УПК. С 1988/89 учебного года у нас развивается новая специализация — делопроизводство на базе ЭВМ. Уже со второго полугодия IX класса предполагается разделение по двум специализациям в соответствии с желаниями и возможностями учащихся. На занятиях специализации «делопроизводство» должны преобладать типичные, чисто пользовательские задачи (работа с редактором текстов и принтером, информационно-поисковыми системами, электронными таблицами, другими прикладными программами). Предусматриваются также курсы традиционного делопроизводства (типы документации, правила оформления бумаг, организация канцелярского дела и т. д.). В общем учебный процесс на этой специализации нетрудно себе представить; ее актуальность в доказательствах не нуждается.

Заканчивая рассказ об организационной части процесса обучения в УПК, отметим, что летом в конце IX класса все учащиеся проходят производственную практику на предприятиях, располагающих вычислительной техникой, или в ГКЦ. В последнем случае дети в основном занимаются созданием ППС. Начиная со второго полугодия X класса учащиеся работают над так называемыми итоговыми заданиями, темы которых зависят от специализации, но обязательно группируются вокруг ППС-ориентированного ядра. По окончании курса обучения все учащиеся сдают теоретический и практический экзамены. Итоговая оценка проставляется в их аттестате дважды: по информатике (курс информатики является подмножеством курсов обеих специализаций УПК) и по трудовому обучению.

Теперь опишем процесс сотрудничества учащихся УПК и их преподавателей в создании ППС. Создание сценария и контроль за единообразием интерфейсов — прерогатива преподавателя. Коллективная разработка программ по жестким спецификациям, а также схема «преподаватель структурирует, ученик реализует процедуры» у нас используются мало. Чаще либо несколько человек делают одно и то же ППС, соревнуясь в качестве интерфейса, эффективности и даже элегантности, либо применяется система конвейера: учащийся анализирует программные полуфабрикаты, «импровизации на тему сценария», написанные его предшественниками в разработке данного ППС, и передает уже «обогащенную» своим вкладом «программную руду» следующему разработчику. Технология конвейера — вещь, конечно, вынужденная. Нам приходится считаться с одной существенной особенностью мотивации школьников — они часто работают хорошо только до тех пор, пока им интересно. Эта особенность может быть изолирована первой технологией, однако организация предусматриваемого ею соревнования не всегда возможна. Надо сказать, что поиск ошибок, нерациональных подходов и других недостатков чужой программы — одно из любимых занятий наших учеников, и мы стараемся этим пользоваться. Однако существуют в работе школьников негативные моменты, бороться с которыми довольно трудно. Сюда относятся кроме «неправильной» мотивации (самое неприятное следствие которой — недоделанные программы) стремление к усложненности, вычурности программ и их интерфейсов, а также весьма неразвитая способность поставить себя на место пользователя.

Процесс создания ППС обеспечивается и такими формами обучения, как факультатив, который посещают все желающие и на котором изучаются самые разные вещи — от структуры игровых программ до психологии восприятия изображения на экране, причем докладчиком часто является школьник; на семинарских занятиях мы «проходим» интересные книги (Кнут, Вирт, Громов, «Комбинаторика для программистов» В. Липского и др.) и статьи (из «Кванта», «В мире науки» и др.). Школьники часто приходят поработать на ЭВМ просто в свободное время (правда, у них свободного времени почему-то всегда больше, чем у нас машинного).

В заключение — коротко о результатах нашей работы и планах на будущее. Прежде всего заметим, что хотя причиной создания ГКЦ как единого комплекса разных форм обучения был дефицит техники, скоро выяснилось, что такая организация обучения близка к оптимальной, поскольку обеспечивает достаточно глубокую дифференциацию учащихся (за счет переходов из одной специализации УПК в другую и из группы информатики в группу УПК и наоборот даже в течение учебного года); широкую сферу приложения и практических испытаний программного обеспечения, включенность программиста в учебный процесс, естественность

сопровождения и модифицирования программ; единую идеологию обучения в УПК, ШЮП и на курсе информатики (один и тот же преподаватель ведет разные курсы); возможность объединить в факультативной форме обучения учеников разных групп и возрастов; все преимущества умеренной централизации, особенно в условиях дефицита техники.

Педагогическое программное обеспечение, используемое в ГКЦ, почти полностью сделано школьниками по сценариям и в соответствии с идеологией разработок пединститута и наших преподавателей. Вот наиболее качественные и широко используемые программы, сделанные детьми.

Имитаторы исполнителей: ЧЕРТЕЖНИК, РОБОТ-МАНИПУЛЯТОР (описаны в [1]), МАШИНА ПОСТА, БЕЙСИК (позволяет «следить» за работой «интерпретатора» Бейсика на примерах коротких Бейсик-программ), ПРОЦЕССОР (наглядно демонстрирует работу процессора под управлением машинных команд), ПСЕВДО-SCP (бездисковый имитатор подмножества встроенных команд системы).

Тренажеры: КЛАВИАТУРА, СИСТЕМЫ СЧИСЛЕНИЯ, ШЕСТНАДЦАТЕРИЧНАЯ АРИФМЕТИКА, ЛОГИЧЕСКИЕ ЭЛЕМЕНТЫ, ТИПЫ ДАННЫХ (языка Паскаль).

Тесты по редактору текстов WM и наиболее употребительным системным программам.

Обучающая программа АРХИТЕКТУРА (модель работающей под управлением операционной системы микро-ЭВМ с возможностями «наблюдения» и «вмешательства в работу» «процессора», «памяти», «внешних устройств»; «подключения» своих «устройств», создания соответствующих «драйверов» и т. п.).

Планируются создание обучающих программ по операционным системам, отдельным программным средствам (транслято-рам, текстовым редакторам, программам обслуживания дисков и т. д.), а также по физике, химии, биологии; реализация исполнителей ЛОГИЧЕСКИЙ КОНСТРУКТОР, БАЗА ДАННЫХ; разработка комплексов тренажеров и тестов по различным темам курсов УПК, информатики, ШЮП и школьным предметам.

В области организации и идеологии обучения наша стратегическая цель — создание курса «Теория и практика разработки и применения ППС», некоторые элементы которого уже существуют и описаны выше. Требует существенной доработки программа ШЮП, особенно второго года обучения.

Материально-технической, вернее — экономической, стороне учебного процесса, умолчать о которой нельзя, посвящено

Лирическое отступление 2. Что делать?

Прежде всего хочется выразить полную солидарность с превосходной статьей С. Пачикова [3] и попытаться адаптировать высказанные в ней идеи к проблемам школьной компьютеризации. Главная идея такова: если мы не можем стать страной лучших в мире машин (а в этом уже почти никто не сомневается), давайте хотя бы станем страной лучших в мире программистов. Мировой опыт показывает, что чем раньше начинается обучение программированию, тем больше вероятность стать хорошим (а без этого не станешь лучшим) программистом. Итак, стране нужны лучшие в мире программисты, а школе, где они будут обучаться, нужны лучшие в мире ППС, потому что просто хорошие ЭВМ, даже если их достаточно много,— это еще далеко не все, что нужно для компьютеризации. В странах, где компьютеризация школы технически осуществлена, ожидаемого резкого скачка в качестве обучения, как известно, не произошло. Вывод очевиден — нужно растить лучших в мире программистов, создавая им условия для разработки лучших в мире ППС. Талантливые дети у нас еще не перевелись. Единственное препятствие — качество «железа».

Школьные КВТ (если они есть) заполнены БК, «Агатами» и не намного более передовыми «Ямахами» и «Роботронами». Создав независимо друг от друга три имитации уже устаревшего IBM PC под разными названиями, но одинаково плохо совместимые с прототипом, наши электронные министерства избавили нас от надежды на то, что отечественная промышленность в обозримое время сделает что-то хорошее для школы. А ведь школе нужно не просто хорошее, а лучшее, и не в обозримом будущем, а прямо сейчас. Поэтому, если мы хотим иметь эффективно работающую систему генерации ППС высокого класса (похоже, что именно в этом, несмотря на простоту формулировки, суть компьютеризации школы), придется покупать машины за границей. Другого выхода просто нет.

Где же школа возьмет деньги на покупку «железа»? Ответ прост — заработает. Владельцам многочисленных небольших, но хорошо компьютеризированных фирм нужно показать, что мы готовы работать в два раза больше, а получать в два раза меньше. Нужно создавать многочисленные небольшие группы программистов и посылать их на работу в эти фирмы на короткое время для выполнения разовых заказов. Как составлять такие группы? Из способных школьников VI—X классов, мечтающих о поступлении в университеты и пединституты на «педагогическо-информатические» специальности; таких же

вчерашних школьников, уже ставших счастливыми студентами; их школьных и вузовских преподавателей. Заодно мы сможем продвинуться и в решении главной задачи, стоящей перед школой. Ведь если для успеха компьютеризации нужна эффективно работающая система генерации ППС высокого класса (ясно, что ППС высокого класса предполагают соответствующую технику, организацию и методику), то для успеха перестройки школы вообще нужна эффективно работающая система генерации высокого класса учителей.

У нас есть и менее лирическая программа-минимум, реализация которой в большей степени зависит от нас самих. Мы хотим как можно скорее сделать следующее:

добиться для коллектива преподавателей и учащихся ГКЦ прав трудового коллектива в смысле Закона о государственном предприятии;

окончательно отрегулировать финансовую схему взаимодействия с заказчиками и вопросы оплаты труда учащихся;

обеспечить свободный доступ всех школь-

УПК. Специализация «программирование»

Неделя	Теория	Практика
	IX класс	
	Введение.	Тренажер «Клавиатура» (4 ч)
1	Знакомство с УПК и организацией обучения (2 ч)	
2	История развития и перспективы информатики. Поколения и типы ЭВМ (2 ч)	Знакомство с текстовым редактором WM в экранном режиме (4 ч)
3	Команды поиска и замены в редакторе WM. Работа с текстовым буфером (2 ч)	Практическая работа с редактором (4 ч)
4—6	Структура вычислительных систем: «железо» и программы. Архитектура ЭВМ. Понятие операционной системы. Прикладные программы. Базы данных, их типы. Электронные таблицы (6 ч)	Работа с учебной информационно-поисковой системой и учебной электронной таблицей (12 ч)
7—8	Представление информации в ЭВМ. Системы счисления (4 ч)	Работа с тренажерами по системам счисления (8 ч)
9—10	Графические возможности принтера. Их реализация с помощью редактора WM (4 ч)	Построение рисунков на принтере (8 ч)
11—12	Логические основы ЭВМ. Логические элементы (4 ч)	Работа с тренажерами по логическим элементам (8 ч)
13—16	Алгоритмы и исполнители. Достижимые цели. Ветвления, циклы, подпрограммы (8 ч)	Решение задач для исполнителя ЧЕРТЕЖНИК (16 ч)
17—19	Бейсик. Основные конструкции (6 ч)	Решение задач на языке Бейсик (12 ч)
20—23	Моделирование. Этапы решения задач на ЭВМ (8 ч)	Решение задач с составлением математических моделей (16 ч)
24—27	Вычислительная математика: традиционно «машинные» методы решения уравнений, систем линейных уравнений, вычисления площадей (8 ч)	Программирование методов вычислительной математики (16 ч)
28—32	Паскаль. Основные конструкции. Типы данных. Процедуры и функции. Метод последовательной детализации (10 ч)	Решение задач на языке Паскаль (20 ч)
33—34	Теоретический зачет за IX класс (4 ч)	Практический зачет за IX класс (8 ч)
	X класс	
1—6	ОС SCP. Основные системные команды и программы (12 ч)	Работа с основными средствами ОС SCP (24 ч)
7—8	WM. Все команды (4 ч)	Программирование в WM. Тест по редактору WM (8 ч)
9—13	Паскаль. Динамические структуры данных. Возможности системы Турбо-Паскаль (10 ч)	Решение задач на Паскале методом последовательной детализации (20 ч)
14—16	Изучение «настоящей» электронной таблицы (6 ч)	Решение задач учета и оптимизации (12 ч)
17—23	СУБД DATABASE II (14 ч)	Работа с DATABASE II в прямом и программном режимах (28 ч)
24—27	Современные информационные технологии и принципы программирования (8 ч)	Создание различных имитаторов и тренажеров и работа с уже существующими (16 ч)
28—34	Итоговая работа, консультации к экзамену	

ников города к конкурсу в УПК;

обеспечить постоянное обновление машинного парка ГКЦ.

В заключение приведем программу по специализации «программирование» УПК и программу первого года обучения ШЮП. Программа по специализации «делопроизводство» находится сейчас в стадии становления, и приводить ее черновой вариант, видимо, не стоит. Это же относится и к программе второго года обучения в ШЮП, которая должна включать элементы булевой алгебры, комбинаторики, теории графов, методологии нисходящего проектирования и ориентироваться на задачи создания ППС.

Школа юных программистов.
Первый год обучения.

Что умеет ЭВМ? (2 ч).

Алгоритмы и исполнители (4 ч).

Исполнитель ЧЕРТЕЖНИК. Прямой и программный режимы работы. Линейные алгоритмы (6 ч).

Исполнитель РЕДАКТОР (4 ч).

Ветвления и циклы (4 ч).

Ветвящиеся и циклические алгоритмы для ЧЕРТЕЖНИКА (8 ч).

Вспомогательные алгоритмы (4 ч).

Реализация подпрограмм у ЧЕРТЕЖНИКА (10 ч).

Данные. Табличная организация данных (4 ч).

Исполнитель РОБОТ (4 ч).

Составление программ с использованием ветвлений, циклов и подпрограмм для РОБОТА и ЧЕРТЕЖНИКА (18 ч).

Всего 68 ч.

Автор благодарен Н. М. Долматовой, С. В. Сморчковой, Н. В. Оленевой, М. В. Сапиру за ценные замечания по содержанию статьи, а маме и брату Роме — за помощь в редактировании текста.

Литература

1. *Гейн А., Житомирский В., Линецкий Е., Сапир М., Шолохович В.* Программно-методический комплекс по курсу школьной информатики // Информатика и образование. 1988. № 3, 4.

2. *Линецкий Е., Сапир М.* Школа юных программистов: Методическая разработка. Свердловск, 1987.

3. *Пачиков С.* Что же делать? // НТР: проблемы и решения. 1988. № 16.

POLY

В Центре компьютерного обучения при ИАЭ им. И.В.Курчатова специализированный кооператив ПОЛИНОМ по оказанию информационно-вычислительных услуг продолжает заключение соглашений с авторами на коммерческое тиражирование педагогических программных средств для всех типов учебной вычислительной техники (КУВТ-86, КУВТ Корвет, УК-НЦ, Ямаха), рекомендованных экспертной комиссией Государственного комитета по народному образованию СССР.

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

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

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

А. ШПИЛЕВСКИЙ
г. Чебоксары

Фрактальные кластеры

Моделирование береговой линии

«Вначале сотворил Бог небо и землю. Земля же была безвидна и пуста, и тьма над бездною, и Дух Божий носился над водою».

Эта живописная картина, нарисованная в первых стихах книги Бытия, послужит отправной точкой исследовательской экспедиции, которую мы предпринимаем ныне на ковчеге «Берег».

Судно бросило якорь невдалеке от первозданного и пока еще идеально ровного берега, отделяющего землю от воды. Господь, только что закончивший свои труды, не озаботился созданием иссеченной береговой линии в ее современном варианте — с песчаными плесами, скалистыми островами и потайными бухтами. Он, видимо, полагал себя творцом не форм, а законов и, не стремясь к разнообразию, прочертил по линейке прямую, положив по одну сторону от нее сушу, а по другую оставив воду. Находящиеся на ковчеге исследователи присутствуют при историческом моменте. На линии горизонта показались первые волны. За дальностью расстояния точечно видно только их гребни. Гонимые безжалостным ветром, меняющимся ежеминутно, волны мечутся из стороны в сторону, но неуклонно приближаются к берегу. Наблюдатели на судне отмечают, что каждая волна на каждом шаге движения практически случайно выбирает одно из трех возможных направлений к берегу. Этот характер движения волн сохраняется до момента их столкновения с берегом. Волна, накатывающаяся на элемент берега, либо уничтожает его, если ее сила достаточна для разрушения данной береговой породы, либо, отразившись, продолжает движение по тому же закону.

В программе «БЕРЕГ», моделирующей этот процесс, для каждого элемента берега выбирается случайное целое число, лежащее в некотором наперед заданном интервале. Оно является критерием твердости и задает количество волн, необходимых для разрушения данного берегового элемента. Границы всего интервала, таким образом, определяют общий разброс по условной твердости береговых пород. При каждом ударе волны о берег из критерия твердости берегового элемента вычитается единица. Если после этого критерий становится нулевым, волна смывает участок берега, с которым вошла в контакт, и, выполнив свою миссию, исчезает. Вместо «отработавшей» волны на лини горизонта тут же появляется новая. Если критерий не равен нулю после вычитания, волна вновь выбирает одно из трех случайных направлений движения.

Для улучшения динамики работы модели линия горизонта вплотную приближена к линии первоначального берега, и волны, появляющиеся в случайных местах линии горизонта, «обслуживаются» поодиночно, так что в каждый момент времени существует лишь одна из них.

Главной частью алгоритма модели, приведенного ниже, является бесконечный цикл, выпускающий все новые и новые волны.

```
нач
Рисовать первоначальный берег.
```

Кластером (от англ. cluster — гроздь, пучок, группа) в математике называется множество элементов, сгруппированных по какому-либо признаку в одну группу. В данной статье кластер — односвязное множество точек.

Фрактальность же — это негладкость, «изрезанность».

```
Выбрать для каждого элемента берега
критерий его твердости.
нц
  Выбрать случайные координаты волны на
  линии горизонта в качестве текущих коор-
  динат.
  Смыва берега нет.
  нц пока смыва берега нет
    Рисовать волну по текущим координатам.
    Выбрать одно из трех случайных на-
    правлений и вычислить новые координа-
    ты волны.
    если по новым координатам есть берег
      то
        Уменьшить критерий твердости бе-
        рега на 1.
        если критерий твердости берега
        равен 0
          то
            Стереть волну по текущим ко-
            ординатам.
            Стереть берег по новым коор-
            динатам.
            Смыв берега есть.
        все
      иначе
        Стереть волну по текущим координа-
        там.
        Сделать новые координаты теку-
        щими.
    все
  кц
кц
кон
```

Стирание волны и берега производится цветом, выбранным для изображения водной глади.

Любители оптимизировать свои программы могут подумать над тем, как обойтись без большого массива, сохраняющего для каждого берегового элемента его текущий критерий твердости.

Наша программа «ФРАКТАЛЬНЫЕ КЛАСТЕРЫ», моделирующая описанный процесс в несколько более сложной интерпретации, позволяет выпускать до 256 волн одновременно. На рис. 1 приведена характерная береговая линия. Здесь отдельные точки представляют собой движущиеся по направлению к берегу волны. Варьируя твердость береговых пород, можно получать многие разновидности берегов — от песчаных плесов до скалистых фиордов.

Берега, представленные на картинках, имеют «модную» фрактальную структуру, по своим статистическим характеристикам симметричную относительно растяжения и сжатия (желающих ближе познакомиться с фракталами отсылаем к источникам [1, 2]), а описанная модель, с помощью которой они получаются, есть модель агрегации, ограничен-

1

ной диффузией [3]. Ее суть заключается в том, что модельная частица, проникшая (диффундировавшая) в анализируемую среду, блуждает в ней согласно определенным законам до тех пор, пока не столкнется с имеющимся там первоначальным телом (агрегатом). Как только это произошло, частица присоединяется к агрегату, дополняя его. В результате растет новообразование — кластер, структура которого и является обсуждаемым предметом в рамках данной модели.

В данном случае, правда, использована агрегация «с точностью до наоборот», т. е. блуждающая частица, сталкиваясь с агрегатом, не присоединяется к нему, а, напротив, выбивает элемент агрегата.

Удивительно и любопытно, что при помощи простой и даже примитивной модели агрегации, ограниченной диффузией, удается получить наглядное и правдоподобное представление о механизме весьма сложных естественных процессов, не вполне исследованных в современной науке.

Моделирование электролиза

Сделав несколько маленьких изменений в предыдущем алгоритме, можно превратить его в программу «ЭЛЕКТРОЛИЗ», демонстрирующую процесс электролитического восстановления металла. Во время работы программы в верхней части экрана появляются точки, поведение которых аналогично поведению волн в программе «БЕРЕГ». Они так же опадают книзу, испытывая при этом случайные горизонтальные отклонения. Однако это уже не первобытные волны, а ионы металла, «с боем» продвигающиеся к катоду в электролитической ванне. Катод в виде тонкой линии находится в самом низу. Как только очередной ион достигает либо катода, либо поверхности металла, уже осевшего на него, он, восстанавливаясь, прилипает, а вместо него в верхней части

экрана появляется новый, диффундировавший в ванну. При этом на каждом шаге передвижения иона следует проверять наличие соседних частиц в непосредственной близости от него. Если их нет, ион продолжает движение, если же сосед обнаружен — прилипает к нему.

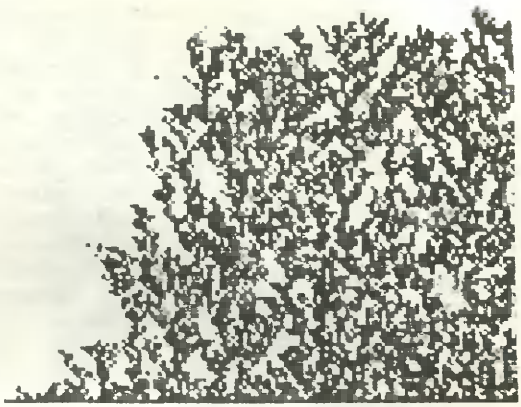
Те, кто захочет поэкспериментировать с этой моделью, могут ввести в свою программу параметры, задающие некоторый интервал горизонтальных координат, внутри которого вылетающие на экран ионы получают свои начальные значения. Тогда можно будет наблюдать кластеры, получающиеся при потоках диффундирующих частиц различной ширины, а также повозиться с катодами различной конфигурации. На рис. 2 представлен электролитический кластер, получающийся при катоде, сжатом до совсем небольшого отрезка.

Отчаянным головам (не боящимся не столько грозящего усложнения программы, сколько падения быстродействия) можно рекомендовать ввести в качестве параметра радиус взаимодействия, увеличивая который исследователь мог бы повышать напряжение на катоде от эксперимента к эксперименту.

Программа «ФРАКТАЛЬНЫЕ КЛАСТЕРЫ», реализующая эти и другие возможности моделирования процесса электролиза, позволяет получить весьма любопытные результаты. Вот только два характерных кластера: для нулевого радиуса взаимодействия — рис. 3; для радиуса взаимодействия в 10 ед. — рис. 4.

Здесь наблюдается все та же фрактальная структура, плотность которой резко падает с ростом радиуса взаимодействия.

2



3

4

С физической точки зрения это понятно: ион, попавший между двумя случайно образовавшимися бугорками, тем с меньшей вероятностью достигает дна ямки, чем больше радиус взаимодействия частиц, так как, скорее всего, он притянется вершиной одного из бугорков. Таким образом, каждый добавляющийся элемент кластера подавляет возможности параллельного роста в зоне радиуса взаимодействия вокруг себя. Если приложить к кластеру линейку, разместив ее параллельно катоду, и считать количество частиц, пересекаемых ею, то станет ясно, что среднее расстояние между частицами примерно равно двум радиусам взаимодействия.

К сожалению, фрактальная модель не дает возможности увидеть, симметризованные, дендритные образования, которые должны были бы появляться при достаточно высоких напряжениях на катоде, когда скорость процесса увеличивается настолько, что анизотропные характеристики металла начинают играть решающую роль в формировании кластера. Если же анизотропия отсутствует,

модель агрегации, ограниченной диффузией, работает вполне удовлетворительно.

Наблюдатели, обладающие воображением, возможно, захотят подумать и об иной интерпретации наших картинок. Изображение на рис. 4 кажется вполне «биологическим». В самом деле, если спускающиеся сверху частицы рассматривать как кванты солнечной энергии, доставляющей участкам деревьев возможность расти, то картина эволюции леса, борьбы деревьев друг с другом, стремления их в высоту становится как нельзя более наглядной.

Модель дымохода

Заменив катод на две вертикальные (или лучше чуть наклонные) стенки, запуская частицы снизу вверх и переименовав их из ионов в крупицы сажи, мы получим уникальную возможность увидеть происходящее в обыкновенной печной трубе. Изображение на рис. 5 (одна из стенок дымохода) способно, кажется, убедить кого угодно в необходимости своевременной чистки труб.

В действительности фрактальная структура сажи в дымоходе представляется весьма вероятной; упоминание об этом можно найти, например, в [3]. Однако желающим добиться здесь большего соответствия реальности придется повозиться с агрегацией «кластер—кластер», так как летящие частицы сажи слипаются вначале между собой в небольшие подвижные кластеры, а уже последние прилипают к трубе или неподвижному кластеру на ней.

По сложности реализации программирование этого механизма достойно любознательных честолюбцев.

5

Моделирование электролиза в поле точечного катода

Чрезвычайно интересно смоделировать электролиз в ванне, где точечный катод расположен в центре окружности, играющей роль анода. Каждый ион в этом случае блуждает по ванне во всех направлениях, пока не агрегирует, коснувшись катода. Допущение о случайном блуждании катода в ванне достаточно реалистично при малых электрических полях. В самом деле, силовое действие катодного потенциала должно ограничиваться некоторым радиусом, далее которого катод оказывается экранированным ионами, уже подошедшими ближе этого радиуса. С уменьшением выбранного потенциала катода величина этого радиуса взаимодействия может становиться весьма малой.

Лобовое решение задачи может оказаться абсолютно безнадежным из-за ничтожной вероятности агрегации для частицы, находящейся на сколько-нибудь значительном удалении от катода и перемещающейся в случайных направлениях. Это принципиальное затруднение можно преодолеть, использовав маленькую хитрость.

Пусть в начальный момент времени наблюдаемая электролитическая ванна находится на таком удалении от экспериментатора, что кажется очень маленькой и содержит лишь небольшое число различимых и практически точечных областей. За каждый такт времени ион, двигающийся в ванне, проходит расстояние от одной видимой области до другой, выбранное в случайном направлении. Поскольку видимых областей немного, вероятность агрегации велика и вскоре восстановленный ион присоединяется к кластеру. По мере того как кластер растет, мы приближаемся к ванне и увеличиваем масштаб ее представления так, чтобы всегда оставалось не слишком много свободных периферийных элементов, в которых может блуждать еще не агрегировавший ион.

В практической реализации это означает, что вокруг начального кластера или катода выбирается прямоугольное окно, верхняя сторона которого несколько выше самого верхнего элемента кластера, нижняя сторона — ниже самого нижнего элемента, левая сторона — левее самого левого элемента, а правая — правее самого правого. В случайных точках периметра этого окна появля-

ются диффундировавшие ионы, двигающиеся затем в случайных направлениях и отражающиеся от невидимых стенок окна. Если присоединившийся ион оказывается выступающим выше прежней самой верхней точки кластера, верхняя сторона окна отодвигается на шаг вверх. Остальные три стороны ведут себя аналогично.

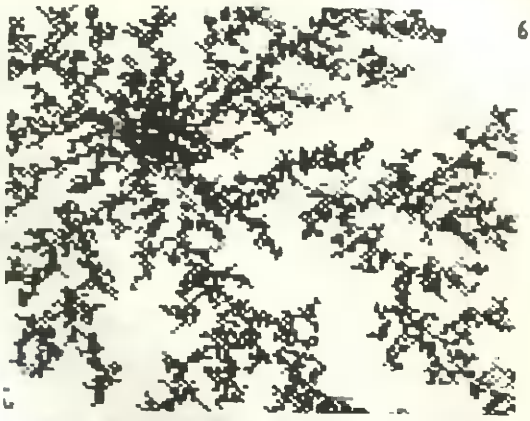
Опишем алгоритм более формально.

```
нач
Рисовать катод.
Выбрать первоначальное окно.
нц
  Выбрать текущие координаты на периметре окна.
  Соседей нет.
  нц пока соседей нет
    Выбрать случайное направление из восьми.
    Считать новые координаты, отражая ион от окна.
    Стереть точку по текущим координатам.
    Рисовать точку по новым координатам.
    Сделать новые координаты текущими.
    Проверить наличие соседей около текущей точки.
  кц
  если новая точка самая высокая
    то
      Верхнюю сторону окна на линию выше.
  все
  если новая точка самая низкая
    то
      Нижнюю сторону окна на линию ниже.
  все
  если новая точка самая левая
    то
      Левую сторону окна на линию левее.
  все
  если новая точка самая правая
    то
      Правую сторону окна на линию правее.
  все
кц
кон
```

Программа «СНЕЖИНКА», выполненная по этому алгоритму, наградит зрителя удивительными фрактальными снежинками (рис. 6), структура которых определенно наводит на философские размышления.

Наиболее основательные экспериментаторы могут и здесь ввести изменяемый радиус междучастичного взаимодействия, наподобие рекомендованного в программе «ЭЛЕКТРОЛИЗ», или поколдовать над хитростями агрегации «кластер—кластер».

Между тем в [3] приводится фотография реального цинкового кластера, появившегося в результате электролиза. Читатели могут сами сделать сравнение действительности с ее моделью.



6

Прочие модели

Описанный механизм агрегации, ограниченной диффузией, является, по-видимому, неплохим представлением многих естественных процессов, и значение его в естествознании еще должно уточняться. Укажем на возможность моделирования в рамках данного подхода различных электрических разрядов, образования речного русла, формирования облаков, эволюции некоторых биологических объектов (ветвей деревьев, кораллов и др.). Читатели могут самостоятельно провести исследования границ применимости метода агрегации и его потенциальных приложений. Можно попробовать свои силы, моделируя стекание дождевых капель по оконному стеклу, отражая в своей модели физику процесса; а кого-то, возможно, соблазнит сложная физика молнии в ее, конечно, упрощенном описании методом агрегации.

При этом трудно ожидать, что в каждом случае наша модель даст безукоризненные результаты. Нет никаких сомнений, что и образование береговой линии, и процесс электролиза, и другие естественные события, которые предлагается описывать с помощью данного механизма, в действительности протекают гораздо сложнее. Однако во всех этих случаях агрегация, ограниченная диффузией, позволяет обнаружить и проследить их философскую общность и в то же время получить результаты, вполне удовлетворительно согласующиеся с действительностью.

Программа «ФРАКТАЛЬНЫЕ КЛАСТЕРЫ», моделирующая все описанные процессы, написана на ассемблере ПЭВМ «Агат» и тиражируется нами. *Адрес для справок:* 428017, Чебоксары, ул. М. Павлова, школа № 27, Шипилевскому А. С.; тел. 24-16-22.

Литература
1. В мире науки. 1987. № 2. С. 104—109.
2. В мире науки. 1988. № 1. С. 88—93.
3. В мире науки. 1987. № 3. С. 62—67.

«Рига-микро» из Кохтла-Ярве

Система «Рига-микро» разработана в НИИ физики твердого тела Латвийского государственного университета специально для КУВТ-86.

«Рига-микро» отличается:

привлекательной простотой использования;

широким набором программ по различным общеобразовательным дисциплинам, испытанных в реальных условиях обучения;

великолепным спектром возможностей по программированию учебных материалов.

Центр НТТМ «Мыте» предлагает услуги в приобретении базового набора из 6 магнитных дискет с высококачественным программным продуктом, реализующим все возможности системы «Рига», и полного технического и методического описания применения.

1. Системная дискета содержит операционную систему для ДВК и программу обслуживания сети «Рига».

Внимание!

Новая версия системы «Рига» существенно отличается от предыдущей:

возможно создание экранных копий (с рисунками), сохранение их на дискете и распечатка в графическом виде;

возможно совмещение рисунков, сохраненных на дискете и посылаемых на БК, с диалогом, организуемым преподавателем;

возможна распечатка любых текстов со сменными алфавитами;

повышена надежность сети.

2. Дискета обучения работе в операционной системе содержит программы для приобретения учителем навыков работы с ДВК.

3. Дискета первого контакта содержит программы, формирующие первые навыки работы с БК-0010.

4. Дискета обучения языку Бейсик содержит программы, автоматизирующие процесс обучения Бейсику.

5. Дискета изучения применения ЭВМ предназначена для проведения занятий по использованию ЭВМ при решении наиболее распространенных задач: подготовка текстов, табличные расчеты, ввод и поиск информации.

6. Дискета демонстрации возможностей ЭВМ содержит программы, демонстрирующие возможности ЭВМ при обучении физике, химии, биологии и др.

По вашему желанию Центр НТТМ «Мыте» может сформировать дискеты с конкретным набором программ по различным учебным дисциплинам (физика, русский, английский, немецкий языки, военное дело и др.), а также для конкретного исполнения КУВТ-86 (с языком Фокал или Бейсик).

Оплата производится по договорной цене.

Стоимость базового комплекта — 2000 рублей.

Центр НТТМ «Мыте» проанализирует Ваши потребности и цели, разработает программы, соответствующие Вашим требованиям, обеспечит дальнейшее сопровождение всех услуг.

За более подробной информацией обращайтесь:

202020, ЭССР,

г. Кохтла-Ярве, РУС а/я 15, центр НТТМ «Мыте».

Телефоны (код 01433): 26-510, 49-781.

Журнал в журнале
для школьников, студентов,
учащихся СПТУ и техникумов

Издается при участии ЦК ВЛКСМ

4

МОЛОДЕЖНАЯ ИНИЦИАТИВА

В НОМЕРЕ:

О Всесоюзном конкурсе «Юный программист»

Осенью 1988 г. Вычислительный центр СО АН СССР (г. Красноярск), Красноярский педагогический институт, Центр НТТМ «Спец» через журналы «Информатика и образование» и «Квант» объявили Всесоюзный конкурс среди школьников по созданию обучающих, игровых программ, инструментальных средств для детей до 12 лет.

Конкурс преследовал несколько целей.

Во-первых, он должен был заполнить возникший вакуум идей и, быть может, программных продуктов в этой довольно узкой области, содействовать популяризации их создания в дальнейшем. В СШ № 41 г. Красноярска в течение трех лет ведется экспериментальная разработка содержания компьютерного образования в младших классах, поэтому мы были самыми заинтересованными участниками, зрителями, судьями.

Во-вторых, мы хотели дать возможность на равных поучаствовать в конкурсе как начинающим, так и почти профессионально работающим ребятам.

В-третьих, что может быть лучше для воспитания детей, чем помощь старшего младшему! Причем надо не только придумать и написать хорошую программу, но и внедрить ее. А это значит, что автору приходится сопровождать программу на уроках, смотреть, как она воспринимается, разговаривать с малышами, помогать им, внося исправления, делая свой продукт все лучше и лучше. Этот процесс довольно длителен и полезен для старшего не менее, чем для младшего.

Участникам конкурса нужно было прислать подробно описанные сценарии своих программ.

Первого февраля 1989 г. закончился срок их приема. Были подведены итоги I тура.

Приведу некоторые статистические данные. В конкурсе приняли участие 166 школьников из 62 городов и поселков 8 республик нашей страны, 7 болгарских школьников.

Наибольшее число заявок пришло из Москвы (15), Ленинграда (13), Алма-Аты (11), Красноярска (9). Девять заявок пришло из маленького поселка Протвино Московской области. Из 166 участников — 54 десятиклассника, 51 девятиклассник, 32 вось-

миклассника. Есть семиклассники, шестиклассники и т. д. до IV класса. Самым юным участником конкурса стал ученик II класса школы № 86 г. Красноярска Паша Москвин.

Популярными оказались машины БК-0010 (55 работ) «Ямаха» (47), «Агат» (21).

Жюри, состоявшее из специалистов разных городов страны, оказалось в сложном положении. Уровень конкурсантов (как программистов) был достаточно высок, но качество сценариев большей частью низко.

Ребята много смотрят, читают, знают, но мало думают. Часть работ, видимо, не была показана взрослым. Либо некому показать, либо не хочется показывать, что одинаково плохо. Но и то, что было сделано по заказу,— неудовлетворительно. На этом фоне очень хорошо смотрелись работы детей из Дворца пионеров Алма-Аты, болгарских школьников.

Жюри отобрало 60 программ для участия в финале. Финал конкурса состоялся 1—4 апреля 1989 г. в г. Красноярске. К сожалению, не все участники смогли к нам приехать.

Финал проходил в трех разных местах, в зависимости от типа компьютера, по очереди, в течение трех дней.

Организация была такова: каждому участнику была выделена машина и в течение нескольких часов он показывал жюри и другим участникам свою программу. Кроме главного жюри параллельно работало жюри зрителей.

Работы были представлены очень разные.

Целый пакет тестов на логическое мышление, реакцию, запоминание привезли ребята из Башкирии, интересную программу для тех, кто хочет научиться фотографировать, создал восьмиклассник Дима Коваль из СШ № 31 г. Челябинска, самообучающуюся программу игры «Реверси» привез девятиклассник из СШ № 12 Ленинграда Вадим Бакман, программу, позволяющую кроить юбку по индивидуальным размерам, показала десятиклассница из СШ № 521 Ленинграда Арина Пожидаева.

Сначала — о наиболее интересных работах.

Сашей Карпиком из СШ № 6 г. Барановичи Брестской области был создан оригинальный клавиатурный тренажер. Экран дисплея делится на 13 окошек. В каждом из них расположены две буквы, одна — из левой части клавиатуры, другая — из правой. Играют двое. В одном из окошек случайным образом загорается лампочка. Игрокам надо как можно быстрее найти и нажать клавишу с ближайшей к ним на клавиатуре буквой из этого окошка. Машина фиксирует первое правильное нажатие и считает очки.

Ряд любопытных находок жюри увидело в обучающих программах по арифметике, русскому языку.

Девятиклассники из СШ № 25 г. Иркутска Алексей Волчатов и Денис Голубев создали программу-тренажер по арифметике (I—IV класс). Наряду с примерами типа: 5+3=? они использовали и другие построения: ?×5=20 или 7—?=3.

Николай Бутвин из СШ № 11 того же Иркутска в подобной программе для совсем маленьких приводит такие задачи на сложение и вычитание: + = . В ответ ребенок должен нажать любую клавишу столько раз, каков результат у него получится. С такими примерами можно работать и не зная цифр.

Любопытную идею игры придумали ребята из СШ № 10 г. Ангарска Игорь Новиков и Олег Апостолов. Гусеницу, состоящую из восьми звездочек разного цвета, надо закрасить в один цвет, двигая ее под дождем из разноцветных капель. Капля, попадающая на гусеницу, окрашивает соответствующую звездочку в свой цвет. Под воздействием ветра в любой момент может измениться направление движения капель.

Две новые игры показал десятиклассник из СШ № 76 г. Ростова-на-Дону Миша Борисенко.

Игра «Лестница» лишь внешне похожа на известную игру «Тетрис». Игровое поле имеет вид длинного стакана, по дну которого бегает человечек. На человечка падают блоки. Двигаясь, человечек должен заставить падать блок так, чтобы из блоков выстраивалась лестница, по которой человечек должен подняться наверх.

Игра «Паром» развивает логическое мышление и реакцию. На экране — множество островов, соединенных друг с другом паромами. На некоторых из этих островов лежат клады. Путешественнику нужно собрать все клады. Ему мешает их хозяин. Специфика переправ на паромах делает игру существенно отличной от похожих игр.

Из показанных лабиринтов можно отметить игру «Маляр» красноярских ребят из СШ № 41 Петра Денисенко и Олега Хвостенко. Маляр должен покрасить пол в лабиринте и вернуться обратно, имея на каждом уровне одно небольшое ведро краски. В некоторых местах пол настолько плох, что его приходится красить дважды. «Маляр» был награжден специальным призом жюри за гармоничность идеи и средств ее выражения.

Было несколько инструментальных программ, но отметить можно лишь идею ретранслирования, осуществленную в реализации мо-

дифицированного языка программирования «ЛОГО» на БК-0010 ленинградским шестиклассником Алешей Эфросом. Картинку, нарисованную непосредственно графическим карандашом, можно представить программой на языке «ЛОГО».

Вот, пожалуй, все самые интересные идеи, которые удалось увидеть в работах ребят. Не густо, но учтем, что подобный конкурс проводится впервые, и вспомним о его специфике.

Теперь поговорим о достоинствах и недостатках.

Бросается в глаза плохое знание русского языка, тексты иногда просто неграмотны. Часто употребляются иностранные слова (особенно в заставках). Текста вообще должно быть мало, но если он есть, то буквы должны быть крупными, хорошо различимыми, текст — отцентрирован, без переносов на экране. Каждая программа должна иметь ясную, лаконичную инструкцию, подсказки.

Клавиши должны соответствовать тем командам, которые надо выполнить. Существуют некоторые стандарты: select — выбор, esc — смена режима, окончание работы; пробел — переход от кадра к кадру; стрелки, ввод имеют естественные значения. При выборе клавиш главным критерием должно быть удобство для пользователя. В программах, предназначенных малышам, используемых клавиш должно быть как можно меньше.

Четвероклассник из пос. Протвино Московской области Виталий Луговский привез на конкурс развивающую игру «Спички». Это была самая красивая программа из всех представленных. Но у нее оказалось столь сложное управление, что в итоге эта потенциально хорошая работа не вошла в 20 лучших. Программу должны были посмотреть взрослые и подсказать ему то, что в силу своего возраста он мог и не знать.

Валера Мартынович из СШ № 25 г. Бреста показал программу «Путешествие в сказочную Страну знаний». Программа имела серьезные недостатки, но отлично продуманное управление. При большом содержании — минимальное количество клавиш, нарисована специальная клавиатура, которая накладывается на пленочную клавиатуру БК-0010. Названия клавиш соответствуют объектам программы. Например, на одной из клавиш нарисована баба-яга на метле.

Неграмотно используется меню, если вообще используется. Должен быть простой выбор, возможность выхода в меню после каждой темы и в конце программы.

Мало и плохо используется окошечная система. А лучше делать так: текст — в одно окошко, график — в другое, подсказка — в третье, картинка — в четвертое. Все должно быть хорошо продумано: число окошек, их величина, цвет, расположение, управление ими и т. д.

Отрадно, что на конкурсе была программа, по своему интерфейсу близкая к идеалу, и все участники смогли ее увидеть.

Программа «Пентамино» ученика СШ № 41 г. Красноярска Миши Белолипецкого получила специальный приз жюри. Миша широко использует систему окошек, меню, различные указатели. Сложная по содержанию программа управляется несколькими клавишами.

Жюри особым образом отмечало отношение программы к пользователю. Специфика конкурса такова, что эта сторона во многих случаях оказывалась решающей. Мы дали специальный приз Олегу Акулову из пос. Тальменка Алтайского края за очень добрый клавиатурный тренажер «Колобок».

Колобок прыгает по мостику через болото с одного берега на другой. Мостик состоит из синих кирпичиков, на каждом из которых нарисован символ. Чтобы пройти через мостик и не упасть в болото, ребенку нужно быстро нажать клавишу с этим символом. Кирпичик становится желтым и не проваливается в болото. Если же Колобок все-таки падает, лягушка, сидящая рядом, начинает радостно прыгать и громко его приветствовать.

Всем очень понравилась исключительно симпатичная, сделанная с любовью к детям программа «В мире объемных фигур» красноярской школьницы Кати Сысоевой с ворохом приятных неожиданностей в середине и необыкновенным концом. Отмечу также, что Катя — начинающий программист, не прошло и года, как она познакомилась с компьютером.

Разработка оригинальных сюжетов, замыслов, тщательный подбор текста, организация управления программой, ее доброжелательность — вот основные моменты, на которые хотелось бы обратить внимание.

И последнее. Лучше всего ребенок усваивает знания, когда он слушает, смотрит, играет, думает, получая информацию об одном и том же самыми разными способами. Поэтому надо стараться использовать все возможности, которые дает компьютер: вывод текста, графику (мультипликацию), игровые формы, музыкальные возможности, но все в меру.

Нужна культура создания подобного рода программ. Этой культуры и не хватало конкурсантам. Этой культуры не хватает сейчас и многим из нас, учителей. Научиться же этому можно, только просмотрев множе-

ство программ, хороших и плохих. Вот такую возможность и дал наш конкурс всем приехавшим.

А теперь о его результатах.

Большой приз (магнитофон «Сатурн») и приз зрительского жюри (сувенирные шахматы) получили болгарские школьники за пакет «Логические игры». Было представлено 26 игр, часть из которых ребята придумали сами.

Первое место занял **Михаил Борисенко** (г. Ростов-на-Дону).

Два вторых разделили: **Михаил Белолипецкий** (г. Красноярск), **Алексей Эфрос** (Ленинград).

Два третьих заняли: **Дмитрий Ланин** (г. Омск), **Екатерина Сысоева** (г. Красноярск).

О трех специальных призах я уже рассказал. Еще 10 участников получили библиотечки книг (20 штук) по информатике и программированию.

За четыре дня ребята успели осмотреть город, съездили на Красноярскую ГЭС, посмотрели первоапрельский КВН и спектакль Красноярского театра оперы и балета «Юнона» и «Авось», увидели красноярские Столбы.

Конкурс решено сделать традиционным. Ровно через год в рамках намечаемой Всесоюзной конференции учителей информатики пройдет детский компьютерный фестиваль, где мы надеемся вновь встретиться со многими ребятами, увидеть их новые работы.

Е. КУЗНЕЦОВ,
Красноярск

Просто, дешево, удобно

Предлагаю перед записью компьютерных программ на магнитную ленту записывать на нее голосом через микрофон ряд чисел, например натуральный ряд: 1, 2, 3, 4, 5, 6 ... и т. д.

Естественно, что сначала должна быть сделана эталонная запись путем обычной диктовки, а затем эталонная запись переписывается на все новые кассеты с пленкой, предназначенные для хранения программ.

Перед записью программы необходимо «пропустить» хотя бы одно число в режиме воспроизведения, т. е. не стирать на маленьком отрезке пленки фонограмму хотя бы одного числа. Практичнее оставить два, три числа.

Эти числа (или одно число) необходимо записать в каталог данной кассеты вместе с названием записанной программы.

Предположим, что на одной стороне компакт-кассеты записано 10 файлов, начинающихся, например, с чисел 2, 104, 200, 250, 320, 637, 699, 718, 811, 867, 895, 936.

Зная, что на этой пленке поместилось ориентировочно 1000 чисел, можно приблизительно перемотать пленку в соответствии с нужным числом, с которого начинается нужный файл, и, «нащупав» с помощью последовательных проб место, где звучит голос с данным числом, произвести загрузку файла в компьютер.

При повторной записи файлов на уже занятую пленку необходимо ее снова разметить.

Преимущества предлагаемого способа поиска файлов на магнитной ленте очевидны.

В. МАЛЬКОВ,
СШ № 555 г. Москвы, IX класс

Гонорар за эту заметку автор передает во Всесоюзный детский фонд им. В. И. Ленина.

Соревнуются программисты

В апреле 1989 г. на базе Рижского техникума радиоэлектронного приборостроения проводился второй — заключительный — этап конкурса на звание «Лучший программист» между техникумами Министерства промышленности средств связи. Первый этап был проведен в период с сентября 1988 г. по март 1989 г., около 400 учащихся соревновались между собой внутри техникумов. Во втором этапе конкурса — межтехникумовском — приняло участие 39 учащихся из 16 техникумов отрасли.

Конкурс проводился по двум группам. В первую входили учащиеся, будущая специальность которых связана с вычислительной техникой. Во вторую входили учащиеся других специальностей.

Тематика конкурсных работ была обширна. Это пакеты прикладных программ (ППП) для проведения расчетов по курсовому и дипломному проектированию; ППП, предназначенные для использования при выполнении лабораторных и практических работ; обучающие системы и контролирующие курсы, отдельные подпрограммы систем АСУ, САПР и микропроцессорной техники; игровые и тестовые программы.

Работа на ЭВМ велась с 9 до 21 ч в течение четырех дней. Большую организатор-

скую работу проделал начальник УВЦ Рижского техникума радиоэлектронного приборостроения Д. П. Кукайнис.

На заключительном торжественном заседании председатель конкурса начальник УВЦ Ленинградского радиоаппаратостроительного техникума Е. А. Неймарк зачитал протокол, в котором были названы победители. Ими стали в первой группе: О. В. Косвинцева, Ленинградский ТРП, 1-е место; С. В. Датий, Рижский ТРП, 2-е место; А. Я. Сидоров, Рижский ТРП, 3-е место. Во второй группе места распределились так: М. В. Зрелов, Ленинградский РАСТ, 1-е место; В. К. Кураш, Рижский ТРП, и Д. В. Аристархов, Ленинградский РАСТ,— 2-е место; Ю. В. Неволин, Артемовский ТТП, Ю. В. Колмоков, Московский РТТ, и С. Н. Кузнецов, Ростовский ТРП,— 3-е место.

Опыт проведения конкурсов программных средств показал, что техникумы отрасли успешно справляются с поставленными задачами в области внедрения новых информационных технологий обучения. Участники конкурса показали хороший уровень знаний языков программирования, умение творчески мыслить, способность быстрой адаптации в операционной системе. Все программы участников имеют практическую ценность. В ходе конкурса прошел полезный обмен опытом работы на различных СВТ.

Представители техникумов высказались за проведение таких конкурсов не только в отрасли, но и в более широком масштабе.

Следующий подобный конкурс в системе техникумов Министерства промышленности средств связи будет в 1991 г.

Н. ДОВЫДЕНКО

Дневник

Этот дневник я нашел сегодня у себя на столе. Непонятно, каким образом он у меня оказался. На обложке было написано большими буквами: «Быстро перепиши и передай этот дневник тому, кто собирается делать самопрограммируемые машины».

20.07.2021 г.

Не знаю, почему я решил завести этот дневник. Мне уже 17 лет. Я работаю в Центре робототехники. О своей работе писать не хочу, потому что там не всё в порядке. Я живу в 30-этажном железопластиковом доме (в XX в. применяли железобетон, но такие дома были признаны нерациональными; железобетонные дома были холодными и искажали радиоволны). В моей квартире четыре комнаты: гостиная, спальня родителей, кабинет отца и моя небольшая комната. На столе в моей комнате стоит то, чем я больше всего горжусь. Это компьютер последней марки с видящим блоком. Их выпустили всего 10 штук, и отец принес мне один, взяв с меня слово, что я буду хорошо работать и не буду устраивать конфликтов с начальством.

22.07.2021 г.

Вчера я не мог сделать запись в дневнике. Было много дел. Мы наконец приступили к конструированию самопрограммируемого робота. Основная идея, которую мы положили в основу проекта: самосовершенствуясь, робот может поставить себе задачу или цель, а затем выбирать наилучший путь ее достижения.

24.07.2021 г.

Вчера целый день работал со своим компьютером. Он показывает феноменальные вещи. Я связался с Банком информации и каким-то образом сумел подобрать ключи к запрету на доступ к информации. Со старым компьютером я мыкался два года, чтобы открыть всего один запрет.

25.07.2021 г.

Сегодня я много работал над пространством и временем и вычислил, что все-таки можно перемещаться в прошлое, но только на небольшие расстояния. С моей не очень-то мощной энергетической сетью я, наверное, смогу переправить только небольшие предметы.

26.07.2021 г.

Мы собрали 10 самопрограммируемых роботов и отправили их на испытания. Меня посылают в командировку следить за проведением испытаний и поведением роботов в экстремальных ситуациях. Улетаю сегодня.

1.08.2021 г.

Наши роботы отлично прошли испытания, и их уже начали внедрять в производство. В основном их закупила армия. Они будут следить за ядерными базами, контролировать запуск ракет в случае войны, но я надеюсь, что войны не будет.

3.09.2021 г.

Все реже стал возвращаться к своим записям. Целыми днями сижу за компьютером. Уже могу переправлять на 20 лет в прошлое вещи величиной 2×3 см, только назад они пока не возвращаются.

10.09.2021 г.

Вчера отправил в прошлое кусок стекла величиной в тетрадь, а сегодня его вернул. Он совершенно не изменился, только прибыл каким-то мокрым. Наверное, угодил в лужу или под дождь.

12.09.2021 г.

Я научился контролировать отправку вещей в прошлое. Уже могу отправлять их точно в определенный год XX в., но они должны быть не больше тетради. С живыми существами пока ничего не получается. Я отправил змею, но она вернулась не живой, а каменной. Точно такой же, как я ее отправил, но из камня.

14.09.2021 г.

Большие неприятности на работе. С роботами стало происходить что-то неверо-ятное. Когда проводился профилактический осмотр одного из наших «детищ» и распечатали содержимое его памяти, то всем стало жутко. Этот робот собирался разнести одну страну, но почему именно эту никто из нас не может понять.

20.09.2021 г.

С роботами явно что-то случилось. Они не отвечают на запросы. Я пытался прорваться в их память с помощью моего компьютера, но ничего не вышло.

30.09.2021 г.

Это ужас! С ядерных баз начали вылетать ракеты, направленные на наши космические колонии, где живут и работают тысячи людей. Мы пытаемся прорваться к роботам, но они заварили все входы.

15.10.2021 г.

Наверху бушует ядерная зима. Я знаю, мы не выживем долго. За пять дней мы вырыли и оборудовали бункер. Что-то мы не учли, создавая самопрограммируемые машины. Они преступили закон: «Робот не должен принести вреда людям». В результате ошибки у них теперь одна цель — уничтожить нас.

Мне удалось пронести с собой в бункер компьютер и энергетический блок, с помощью которого я смогу переправить этот дневник в прошлое.

16.10.2021 г.

Сегодня я делаю последнюю запись в дневнике и отправляю его в прошлое. Я прошу, прежде чем кто-либо соберется создавать самопрограммируемые машины, продумайте все возможные варианты программ и не применяйте их в военных целях. Они расшифровали все ключи, запрещающие доступ, и устроили войну, которую не выиграл никто: ни они, ни мы.

Я отправляю этот дневник в конец 80-х годов, в самый расцвет компьютерной техники. Того, к кому он попадет, я прошу выполнить то, что написано на его обложке. Прощайте!

АРТЕМ САМОЙЛОВ,
СШ № 5, IX-Б,
Москва

Кооператив
«ЭЛЕКТРОН»

предлагает
владельцам ПЭВМ ДВК, БК-0010,
РК-86, «Микроша», «Специалист», «Спектрум»,
«Агат», УКНЦ широкий выбор системных,
прикладных, игровых, учебных программ.

Заключаем
с авторами договоры
на тиражирование
разработанного ими
программного обеспечения.
Предприятиям, учреждениям,
учебным заведениям
оказываем содействие
в реализации и приобретении ЭВМ всех типов.

Адрес
для запросов
каталогов
и условий договоров:
103489, Москва, корп. 705,
кооператив «Электрон».
Телефон 536-12-81
по вторникам, четвергам
и субботам с 12 до 20 ч.

Копирование без проблем

Сегодня, когда число написанных для БК программ приближается к тысяче, никому не приходит в голову самостоятельно составлять все программы, необходимые для работы или для развлечения,— проще воспользоваться готовыми. Мы уже обращались к вопросу о том, где брать эти программы (см.: Информатика и образование. 1988. № 6). И все же сегодня основным способом попрежнему остается перезапись программ у знакомых.

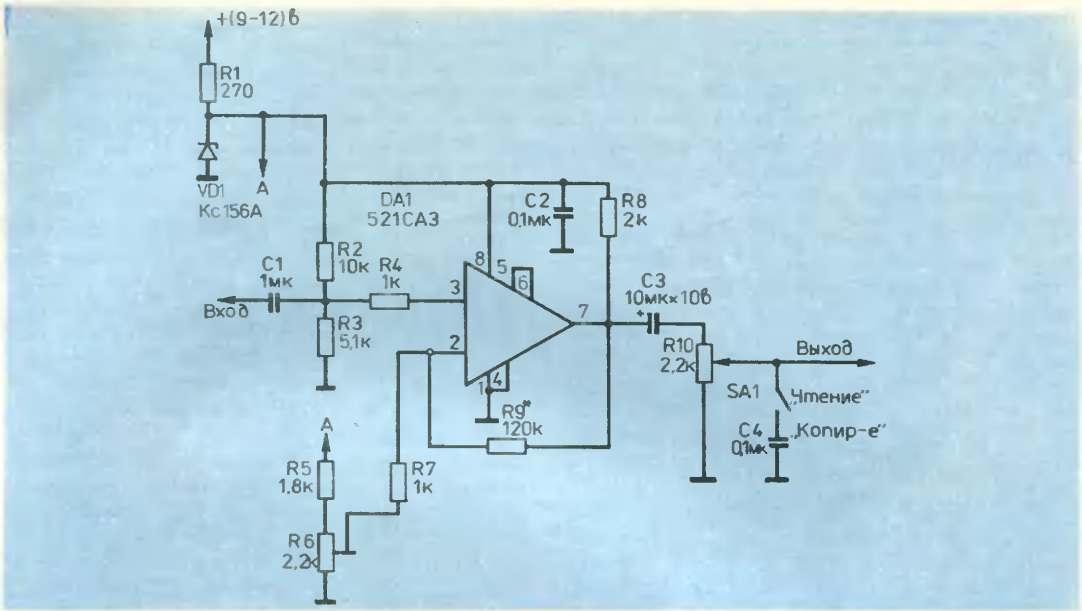
Стандартный способ — программа считывается с магнитофона в память компьютера, затем ее переписывают на другую кассету. Если речь идет о сотне программ, то копирование превращается в долгое, нудное мероприятие, где не обойтись без того, что какие-то программы окажутся пропущены или будут записаны с ошибками.

В последнее время все чаще приходится сталкиваться с тем, что более опытные пользователи защищают свои программы от перезаписи, используя для этого различные ухищрения. Другие опытные программисты создают специальные программы-копировщики, которые позволяют переписывать даже те программы, которые снабжены защитой от перезаписи. В свою очередь появляются программы с еще более сложной защитой, которые не поддаются типовым копировщикам.

Такая борьба программ, по-видимому, может развиваться еще очень долго. В то же время для копирования математического обеспечения для БК необходим другой, более эффективный и универсальный способ. Такой, который бы не «обращал внимания» на тип копируемой программы — язык ли это программирования, игра или набор данных, а также не требовал бы вмешательства оператора. Иными словами, необходим копировщик-автомат. Если для обычного владельца БК такой автомат — это удобство и экономия времени, то для клуба БК он просто незаменим.

Такой автомат пытались создать еще первые счастливые обладатели БК. Самый простой вариант — переписывать программы с одного магнитофона на другой, минуя БК. К сожалению, этот вариант, успешно работающий с магнитофонными вариантами программ, например «Коммодора 64», для БК в большинстве случаев не годится — суммарная погрешность двух аналоговых устройств, каковыми являются соединенные кассетные магнитофоны, вносит в программу ошибки. В то же время роль компьютера, который в данном случае является как бы промежуточным буферным звеном, восстанавливающим исходный цифровой сигнал из аналогового и препятствующим тем самым суммированию погрешностей магнитофонов, вполне может выполнить менее сложное устройство. Такое устройство — регенератор цифровых сигналов БК разработали энтузиасты БК из Львова П. Чирков и Р. Булыч. Они не только изготовили и испытали это устройство, но и решили поделиться этой важной информацией с читателями нашего журнала. Прежде чем передать им слово, скажем только, что клубы и кооперативы, которым приходится тиражировать программное обеспечение в большом объеме, смогут поэкспериментировать с этим или подобным устройством и попытаться достичь большей скорости качественного копирования программ. Надеемся, что экспериментаторы, первыми добившиеся успеха, напишут нам о своем способе, а мы сможем познакомить с ним всех читателей.

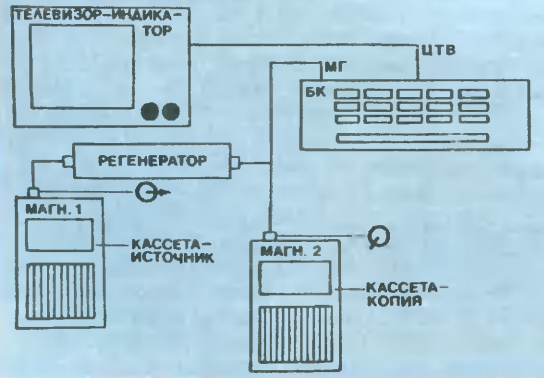
Основной элемент схемы — интегральный компаратор напряжения К521СА3. Эта микросхема обладает двумя подходящими качествами — повышенной точностью, с которой оценивается входное напряжение, а также возможностью работы в режиме однополярного питания +5 В. Пятивольтовое питание позволяет отказаться от применения отдельного блока питания, подключая схему к БК или к одному из двух магнитофонов. По принципу работы устройство является аналогом известного в радиотехнике триггера Шмитта. Как и положено триггеру, триггер Шмитта имеет два устойчивых состояния — назовем их 0 и 1. Триггер обладает гистерезисом, говоря попросту: если плавно увеличивать напряжение на входе, то триггер не сразу изменит свое состояние, а лишь после того, как будет превышен некоторый пороговый уровень. И наоборот, если теперь начать столь же плавно уменьшать напряжение, то триггер Шмитта возвратится не сразу, а лишь после того, как напряжение уменьшится до другого порогового уровня. Разницу между двумя уровнями, когда происходит срабатывание триггера



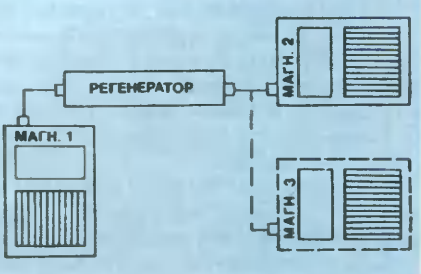
Шмитта, называют шириной петли гистерезиса. Благодаря наличию гистерезисной характеристики в регенераторе осуществляется сравнение входного переменного напряжения с двумя эталонными пороговыми значениями, установленными симметрично относительно нулевого уровня входного сигнала. В результате работы триггера Шмитта входной сигнал, поступающий с магнитофона, преобразуется в последовательность прямоугольных импульсов, т. е. приобретает необходимую исходную форму, которую он имел когда-то на выходе из БК. При необходимости ширину гистерезисной петли триггера можно регулировать с помощью резистора R9. Величина сопротивления этого резистора и ширина петли гистерезиса связаны друг с другом обратно пропорциональной зависимостью.

Настраивают регенератор с помощью осциллографа. На вход устройства подают тональный сигнал звуковой частоты, например, 1000 Гц и амплитудой 0,6 В. На выход схемы подключают осциллограф и, регулируя подстроечное сопротивление R6, добиваются одинакового двухстороннего ограничения синусоиды. Цель можно считать достигнутой, если синусоида превратится в последовательность симметричных прямоугольных импульсов, на языке радиотехников — в меандр. Вторая, и последняя, операция по настройке регенератора — это установка необходимой амплитуды выходного сигнала. Для регулировки служит переменный резистор R10, а установить нужно напряжение с амплитудой, равной примерно 0,15 В. Если регулятор уровня записи магнитофона предварительно выставлен по сигналу компьютера, то указанная амплитуда выходного

Копирование программ. БК выполняет сервисные функции



Тиражирование программ без БК



сигнала схемы должна соответствовать номинальным показаниям индикатора уровня записи магнитофона или немного меньшим. Все вышеописанные операции по настройке схемы следует выполнять при разомкнутом положении переключателя SA1 (положение «чтение»).

Для копирования переключатель SA1 устанавливают в положение «копирование», вход схемы подключают к линейному выходу магнитофона, который работает с кассетой-оригиналом, а выход схемы — к гнезду второго магнитофона, предназначенного для подключения звукоснимателя. Первый магнитофон включают в режим воспроизведения, второй устанавливают в режим записи.

Схема получилась очень простой, поэтому размеры готового регенератора определяются в основном только размерами внешних установочных элементов: разъемов, переменного сопротивления и переключателя. Регенератор получает питание от блока питания одного из магнитофонов (для этого в схему добавлен простейший параметрический стабилизатор напряжения R1, VD1). Стабилизатор позволяет подключать схему к источникам питания с напряжением 9—12 В, т. е. практически ко всем блокам питания современных переносных магнитофонов. Если магнитофон находится на гарантии и открывать его не хочется, регенератор можно подключить к блоку питания БК. Для этого точку А на схеме соединяют с контактами А, В (12), а «землю» схемы — с контактами А, В (2, 3) разъема XT3. В этом случае параметрический стабилизатор не нужен, и цепочку R1, VD1 исключают.

С помощью регенератора можно копировать кассеты с программным обеспечением для БК целиком, независимо от того, что на них записано. Качество копии сравнимо с качеством копируемого оригинала, а нередко и лучше его. Копируя кассету, мы обычно подключаем на выход схемы кроме магнитофона также вход БК, который работает под управлением программы для каталогизации файлов. В этом случае БК играет роль своеобразного индикатора, на экране которого отображается вся необходимая информация о переписываемых файлах. Изготовление устройства не займет много времени, а эффект от его использования оправдает все затраты времени на его настройку.

П. ЧИРКОВ, Р. БУЛЫЧ

Структурный Бейсик для БК-0010

Как известно, Бейсик — это очень старый язык со старой идеологией и, как следствие, даже в своих значительных расширениях (BASIC-MSX, BASICA IBM PC) обладает весьма серьезными недостатками. Основные претензии, которые, как мне кажется, можно предъявить Бейсику,— это отсутствие подпрограмм с именами и параметрами и полная «антиструктурность». Эти недостатки приводят к тому, что программы уже длиной более 100 операторов становятся весьма туманными и слабоуправляемыми, а программы длиной более 500 операторов — вообще недоступными для понимания (а ведь это язык для непрофессионалов!). Таким образом, Бейсик, прекрасный язык для коротких программ, теряет почти все свои прелести при увеличении размеров программы.

Есть и еще один аспект проблемы — учебный. Если посмотреть на процесс обучения информатике в школе, то, с одной стороны, там учат «хорошему программированию» на абстрактном языке, а с другой — тут же ломают «все хорошее», переходя к реальному Бейсику.

Есть ли выход из ситуации? Есть, и, по-моему, очень простой. Нужно ввести в Бейсик средства для структурного программирования (что и делается во всем мире: все разработанные в последнее время версии Бейсика — структурные, с подпрограммами и параметрами). И сделать это можно не только путем создания специальных интерпретаторов и компиляторов, но и относительно несложной модификацией уже существующих. Одна из таких реализаций — для БК-0010 — и предлагается на суд читателей.

Структурный Бейсик для БК-0010 является расширением стандартного MSX-Бейсика, «прошитого» в ПЗУ; он реализуется программой SBASIC, работающей в ОЗУ в адресах 33600—37777. Загрузка и запуск программы SBASIC осуществляются оператором BLOAD "SBASIC", R.

Правила записи операторов в структурном Бейсике такие же, как в обычном, но можно записывать в строке несколько операторов, разделяя их двоеточием (кроме операторов, образующих структуры,— они должны быть единственными в своих строках). Ключевые слова операторов должны зада-

ваться полностью, при этом все имена переменных, включая имена MO, FI и т. п., являются допустимыми. При выводе листинга на экран и на печать текст программы, независимо от исходной формы записи, автоматически форматируется отступами в соответствии со структурой программы. За образец управляющих структур взяты конструкции языка Ада, как наиболее последовательные и наглядные.

Процедуры в структурном Бейсике записываются в виде

```
PROC имя [параметры]
  [описания, комментарии]
BEGIN
  операторы
END PROC
```

Имя процедуры может состоять из произвольного количества символов, из которых значащими являются первые шесть. Имя может содержать русские и латинские буквы, цифры и знак подчеркивания. Передача значений параметрам процедуры осуществляется присваиванием этих значений параметрам при вызове процедуры. (Как и в обычном Бейсике, все переменные, в том числе и параметры процедуры, имеют областью действия весь текст.) Возврат из процедуры осуществляется по оператору END PROC или с помощью оператора RETURN. Вызов процедуры осуществляется в виде

имя процедуры [парам-знач [,парам-знач]...]

Пример. Процедура, рисующая кольцо.

```
PROC КОЛЬЦО X, Y, R, H, C
  'X, Y — координаты, R — радиус
  'H — толщина, C — цвет
BEGIN
  CIRCLE (X, Y), R, C
  CIRCLE (X,Y), R+H,C
  PAINT@(0, R+H/2),C
END PROC
```

Возможные вызовы процедуры:

```
КОЛЬЦО X=120, Y=100, R=60, H=15, C=3
КОЛЬЦО R=20, H=10, C=1 'X,Y — старые
```

Управляющая структура «условный выбор» является достаточно универсальной и соответствует как конструкции «если» алгоритмического языка, так и конструкции «выбор»:

```
IF условие THEN
  операторы
[ELSIF условие THEN
  [операторы ...
[ELSE
  операторы]
END IF
```

Оператор ELSIF, означающий «иначе если», может быть записан в конструкции сколько угодно раз.

Примеры.

```
IF A>0 THEN
  C=A*5+B*3
PRINT A, B, C

ELSE
  C=A*A+B/4
  D=10
END IF
```

```
IF R$="*" AND T$<>"*" THEN
  S=1:PR=1
ELSIF R$="/" THEN
  S=2:PR=1
ELSIF R$="+" THEN
  S=3:PR=2
ELSIF R$="—" THEN
  S=4:PR=2
ELSE
  PRINT «Ошибка в операции»
END IF
```

При необходимости обычный оператор IF в одну строку также может использоваться.

В структурном Бейсике определены три вида циклов: цикл «для», цикл «пока» и безусловный цикл. Цикл «для» аналогичен обычному циклу FOR—NEXT (который также может использоваться).

```
FOR ... LOOP        WHILE условие LOOP        LOOP
операторы           операторы                 операторы
END LOOP            END LOOP                  END  LOOP
```

Безусловный цикл не имеет в заголовке условий выхода. Это или бесконечный цикл, выход из которого возможен лишь при нажатии клавиши (СТОП), либо внутри цикла должен быть записан оператор EXIT. Оператор условного или безусловного выхода из цикла EXIT имеет вид

EXIT [условие]

и может быть записан в теле любого из циклов, кроме FOR—NEXT. Оператор EXIT удобно использовать в тех случаях, когда условие выхода из цикла нельзя проверить в начале цикла; такая ситуация возникает, например, при чтении в цикле по INPUT или по READ.

Пример. Ввод чисел с клавиатуры и запись их в массив MM. Конец ввода — буква «К».

```
FOR J=1 TO 1000 LOOP
```

```
    INPUT «Число»; R$
    EXIT R$="K"
    MM(J)=VAL(R$)
END LOOP
```

Кроме структурных конструкций в структурном Бейсике имеются еще некоторые расширения стандартного языка, заимствованные из других, более развитых версий Бейсика.

Кроме цветов раскраски 1—4 в операторе PAINT можно задавать определяемые программистом цвета 5—16 (см. также: ИНФО. 1989. № 2). Дополнительные цвета определяются матрицей 4*4 точки с помощью оператора

DEF C_n= (цццц, цццц, цццц, цццц),

где n=5—16 — определяемый номер цвета, а ц=1—4 — цвет отдельной точки.

Пример. Раскрасить круг чередующимися в шахматном порядке красными и синими точками.

```
DEF C5=(1313, 3131, 1313, 3131)
CIRCLE (100, 100), 60, 1
PAINT@(0, 0), 5, 1
```

Для воспроизведения музыки служит оператор PLAY следующего формата

PLAY «операнд [операнд] ...»

Операнды в строке записываются непосредственно друг за другом или разделяются запятыми или пробелами. В качестве операндов могут быть записаны имена нот (D, R, M, F, S, L, C), пауза (P), номера октав (O1—O7; первой октаве соответствует O4), темп (T1—T7), задание длительности звучания для нот, у которых длительность не задана явно (H1—H32: от целой до 1/32, по умолчанию 1/4). Вслед за именем ноты может быть указано: длительность (1—32), диез (#), бемоль (—), увеличение длительности в 1,5 раза (.). В операторе PLAY можно записывать не только строковую константу, но и вообще строковое выражение.

Пример. Популярная мелодия.

```
PLAY «T5 O4 MDMDFMR P O3 SSSL8C8
O4 DDD»
```

С. ЗИЛЬБЕРШТЕЙН

Анкета журнала

Уважаемый читатель!
Ваше мнение о журнале в целом или о номере, который Вы держите в руках, редакции не только интересно, но и необходимо. Просим ответить на вопросы нашей анкеты.

Общая оценка журнала (номера) _____

Какие статьи Вас больше всего заинтересовали и почему? _____

_____

_____

Какие статьи не понравились и почему? _____

_____

Печальное приобретение

Хочу поделиться «радостями», которые свалились на меня после того, как наше училище приобрело КУВТ УКНЦ.

1. Класс поставляется в комплекте с операционной системой ФОДОС-2, но никакого описания ее, кроме команды копирования одного диска на другой, нет. Наверное, разработчики документации предполагали, что операционную систему ФОДОС знает каждый преподаватель или что описание ее можно найти в любой библиотеке. Увы, это не так.

2. В документации было сказано, что поставляются языки программирования Бейсик, Паскаль, Макро с описанием. У нас, кроме Бейсика, ничего не было, да и в нем не оказалось команд обмена с диском. Потом, когда я позвонил на завод-изготовитель, выяснилось, что они забыли записать нам еще одну версию Бейсика. Пришлось мне самому ехать в Зеленоград и записывать нужную версию языка.

3. А это уже не неприятность, а беда. У нас не работает сеть, и 12 компьютеров из 13 простаивают, так как в этом КУВТ Бейсик загружается с диска в машину учителя, а затем по сети пересылается в машины учащихся. На заводе-изготовителе («Квант») сказали: «У вас сеть не работает? Это в конструкции мониторов недоработка, и поэтому они вносят в сеть искажения». Хочется спросить: а как конструкторы разрабатывали этот класс? Где же госприемка, ОТК и другие организации и конкретные работники, которые ответственны за качество продукции? Мне известны четыре таких КУВТ, установленных в разных местах, и ни один не работает.

Местное предприятие информатики и вычислительной техники отказалось производить пусконаладочные работы, так как в КУВТ УКНЦ есть конструктивные ошибки. Я не знаю, куда можно обратиться и куда жаловаться. КУВТ, стоящий 32 тыс. рублей, полгода стоит «мертвый», и когда он заработает — неизвестно.

Преподаватель информатики
могилевского СПТУ № 1
Г. БЕРКОВ

Статьи на какие темы и какие новые рубрики Вам хотелось бы видеть в журнале? _____

_____

Из каких периодических изданий Вы черпаете информацию об ЭВМ? _____

_____

Ваши возраст, профессия и место работы? _____

_____

Будем рады, если Вы дополните анкету собственными вопросами и предложениями. Пишите нам по адресу: 107005, Москва, Лефортовский пер., 8. «Информатика и образование». Ждем Ваших ответов!

Взгляд, вооруженный монитором

Наступило время, когда размеры персонального компьютера определяются практически только габаритами видеомонитора, на который выводятся результаты работы программ, системные сообщения и другая алфавитно-цифровая и графическая информация, делающая общение с компьютером не только удобным, но и приятным. А габариты монитора, в свою очередь, определяются разрешающей способностью человеческого глаза и удобством работы с ним.

Американские изобретатели сумели обойти это на первый взгляд непреодолимое ограничение. Они предложили обладателям персональных компьютеров мини-монитор размером со спичечный коробок, который можно прикрепить прямо к очкам или на штативе на уровне глаз. Заглянув в глазок мини-монитора, вы увидите там экран обычного настольного персонального компьютера, стоящего примерно в 60 см от вас, на который можно вывести 25 строк алфавитно-цифровой или псевдографической информации по 80 символов в каждой строке. Кроме отсутствия графики мини-монитор обладает еще одним недостатком — монохромностью. Его цветовая гамма состоит из одного красного цвета, определяемого полупроводниковыми оптоэлектронными приборами, формирующими изображение. Но, видимо, рано требовать большего от монитора, который весит не более 60 г и потребляет меньше полуватта энергии.

Унифицированный вход позволяет подключить мини-монитор не только к различным компьютерам, но и к телефону, телефаксу и видеоиграм, а использование двух мини-мониторов позволяет получать стереоизображение.

Завод на столе

Производство программного обеспечения отличается от других производств. Достаточно один раз вложить силы и средства в разработку программного продукта, чтобы затем довольно долго продавать дискеты с копией исходной программы. Это так просто, что прямо на дому можно открыть завод. Написал программу, и только успевай вставлять в дисковод новые диски. Но что значит — «только успевай»? Каждая такая операция означает, что нужно вставить в компьютер дискету, закрыть дисковод, отдать команду на форматирование, подождать около минуты, пока компьютер подготовит дискету к записи, отдать новую команду и снова подождать, пока программа не будет записана. Вспомним, что обычный компьютер рассчитан на работу с дисководом, но не на тиражирование — может так случиться, что именно дисковод окажется первым устройством, которое выйдет из строя из-за чересчур интенсивной эксплуатации. В то же время, вооружившись «Трейсером» (от англ. trace — след), работу можно упростить и ускорить. Подобно тому как системы «дектоп-паблишинг» позволяют иметь настольную типографию, так и «Трейсер» позволяет открыть домашнюю фабрику по тиражированию программного обеспечения на дискетах. Несмотря на то что дубликатор, а именно так было бы правильно назвать новое устройство, легко размещается на столе рядом с персональным компьютером, он позволяет совершенно автоматически оттиражировать за час 259 дискет для персоналки типа ИБМ. Кстати, 150 чистых дискет можно загрузить разом. «Трейсер» автоматически настраивается на любой заданный формат и с удвоенной скоростью копирует содержимое эталонной дискеты (одновременно с двух сторон чистой дискеты). Работает дубликатор под управлением персональной ЭВМ типа IBM PC / XT или подобной. Если такая производительность кажется вам недостаточной, набор оборудования можно расширить. Подключив к компьютеру еще одно устройство («Трейс 1020» прекрасно разместится под столом), вы сможете вместо одного дубликатора работать сразу с двадцатью. Только успевайте загружать чистые дискеты! Производительность такого микрокомбината — 2880 двусторонних дискет в час. Так и хочется представить подобные комплексы в магазинах-салонах «Электроника». На складе — запас чистых дискет, а в ассортименте — всегда все необходимые программы. Исчерпывается запас какой-то программы — через 5 мин готовы еще 240 дискет.

Разработала и производит новую систему фирма «Трейс», один из участников выставки «Наука-88».

Персональный комбайн

Именно так можно называть теперь персональные компьютеры фирмы IBM, которые можно легко превратить и в «Макинтош», и в «Атари», и в процессор 29000. Осуществлять все эти превращения, после которых на IBM PC AT могут исполняться «чужие» программы, помогают дополнительные платы, эмулирующие архитектуру наиболее распространенных компьютеров. Это, естественно, не исключает возможности использовать компьютер в своем первоначальном архитектурном варианте.

Необходимость создания таких плат вызвана тем, что на мировой рынок постоянно поступает огромное количество персональных компьютеров, лишь часть из которых программно совместима друг с другом. Поэтому обладателям компьютера одного типа недоступны программы, используемые на других ЭВМ. Это вынуждает разработчиков программного обеспечения адаптировать программы на ЭВМ разных типов, вместо того чтобы вкладывать свои знания и опыт в создание новых программ. Эмулирующие платы не только предоставляют потенциальному покупателю большую свободу выбора типа компьютера, но и значительно упрощают перенос уже разработанных программных систем.

Продолжает развиваться принцип модульности, который можно определить словами: «Если тебе темно, не торопись менять дом, попробуй сначала сменить лампочку».

ЧАН ЛЯНТЯНЬ, ВАН ЧЖЭНЬШАНЬ
Институт программного обеспечения АН КНР

Компьютерное образование в школах Китая

До 1982 г. существовало несколько компьютерных кружков, созданных в средних школах, а также в детских центрах технического творчества. Например, работали группы в экспериментальной средней школе, в средней школе при университете Цинхуа и детском центре в Шанхае.

В 1983 г. Министерство просвещения КНР решило провести эксперимент по компьютерному обучению в пяти средних школах при Пекинском университете, а также при университетах Цинхуа и Фудань. Одновременно в Хуадунском педагогическом институте ввели факультативные курсы по теории компьютеров и языку Бейсик. В 1983 г. Министерство просвещения КНР закупило за рубежом компьютеры на сумму 30 млн. юаней. Китайское общество науки и техники организовало молодежные центры компьютерной грамотности в 29 провинциях, городах и автономных районах, предоставив каждому центру по 10—15 компьютеров. С целью повышения у школьников уровня владения компьютерами в 1987 г. Министерство просвещения решило расширить эксперимент, включив в него отдельные школы в Пекине, Шанхае, Тяньцзине, Ухани, Ханчжоу, Гуанчжоу. Таким образом, общее число компьютеров и учеников, изучающих компьютеры, значительно увеличилось (табл. 1).

Таблица 1

Год	Количество, тыс.		
	учеников	учителей	компьютеров
1983	11,8	0,57	0,73
1984	более 20	1,5	10
1985	более 100	3,0	более 50
1988	400	7,0	60—70

Для обучения использовались следующие типы компьютеров: «Эппл II», «Лейсет-310», IBM/PC, «Комекс-35», TRC-80, учебный компьютер CEC-1 («Цзыцзинь»). Распространение компьютеров было неравномерным. Много компьютеров для начальных и средних школ имеется в Пекине и Шанхае. Приведем пример для Пекина (табл. 2).

Таблица 2

Учебное заведение	Типы компьютеров	Количество компьютеров
Экспериментальная средняя школа	«Эппл II»	40
	«Макинтош»	30
Школа № 4	«Эппл II»	50
Средняя школа при университете Цинхуа	«Эппл II»	40
Средняя школа при Пекинском университете	«Эппл II»	50
	«Компак»	10
Средняя школа при Народном университете	«Эппл II»	40
Средняя школа при авиационном институте	«Эппл II»	30
Средняя школа № 161	«Эппл II»	30
Средняя школа Чжунгуанцун	«Эппл II»	50

На занятиях изучаются базовая теория компьютеров, а также языки Бейсик и Лого.

Ныне осуществляется следующий план развертывания компьютерного образования в средних и начальных школах:

1. Установлено обязательное компьютерное обучение для старшеклассников и факультативное для учеников младших классов.
2. Станции детского технического твор-

чества, дворцы пионеров организуют циклы лекций по компьютерам, курсы обучения работе на компьютерах и летние компьютерные лагеря. Это уже делают, например, Пекинский дворец пионеров, Шанхайская станция детского технического творчества.

В г. Тяньцзинь, например, в 1983 г. был создан молодежный компьютерный клуб. Для этого было ассигновано 50 млн. юаней на покупку компьютеров «Эппл II», «Лейсет-301». В настоящее время создано 19 молодежных клубов, которые имеют 600 компьютеров. Под руководством общества по науке и технике была создана сеть компьютерного образования в средних и начальных школах. За 5 лет около 100 тыс. школьников прослушали курсы и прошли обучение на компьютерах. Каждый год на каникулах здесь организуется летний компьютерный лагерь.

Некоторые детские сады и начальные школы уже проводят элементарное обучение работе на компьютерах. Дети с большим интересом используют «Эппл II» и язык Лого и при помощи учителей овладевают основными математическими операциями (например, сложением и вычитанием).

Начальная школа при Пекинском университете открыла факультативный курс по языку Лого. В начальной школе Чжунгуанцун в районе Хайдака имеются кружки, в которых школьники изучают Бейсик.

Для развития компьютерного образования проводятся Всекитайские молодежные компьютерные конкурсы.

Под руководством Госкомитета по образованию и Общества по науке и технике Китайское компьютерное общество ежегодно летом организует компьютерные конкурсы. Раз в год проводятся соревнования по программированию и соревнования по оценке программного обеспечения (эти соревнования чередуются).

Первый Всекитайский конкурс по программированию прошел в 1984 г. в средней школе при университете Цинхуа. В нем приняло участие около 8000 школьников. Участники были отобраны на нескольких этапах из разных провинций и городов Китая. По результатам соревнований 53 школьника получили награды.

Второй Всекитайский конкурс был проведен в 1985 г. в г. Тяньцзинь. В нем участвовало 50 000 школьников. Каждый участник представил собственную программу. Затем участники соревнований сами выставляли программам оценки.

Третий Всекитайский конкурс по программированию был устроен в 1986 г. в Средней школе при университете Цинхуа в Пекине. Были выбраны по три представителя из каждой провинции, города центрального подчинения и автономного района. Участвовали пятнадцатилетние школьники. Сначала проводился письменный экзамен продолжительностью 2 часа (6 задач), а затем проверялось умение работать на компьютере, на что отводился 1 час (5 задач). В 1986 г. ребята, занявшие первые три места и удостоенные первой премии, были в США по приглашению компании «Эппл».

Четвертый Всекитайский конкурс такого типа был устроен в 1987 г. в г. Циндао.

Пятый конкурс по программированию состоялся в 1988 г. в Средней школе при университете Цинхуа. Соревновались пятнадцатилетние и двенадцатилетние (пятнадцатилетние — по языку Бейсик, а двенадцатилетние — по Лого). Участники представляли 28 провинций, городов центрального подчинения, автономных районов, а также Народно-освободительную армию Китая и нефтедобывающую промышленность. Семь учеников получили награды — отечественные учебные компьютеры для начинающих (CEC-1).

Уделяется внимание и обучению учителей.

В крупных городах Китая созданы центры подготовки учителей, которые должны вести компьютерное обучение. Для них ведется преподавание следующих предметов: Бейсик, Лого, работа с ассемблером машины «Эппл». Около 4000 учителей уже прошли такое обучение.

Пропаганда работы с компьютером ведется по нескольким направлениям. Начиная с 1984 г. Китайский детский фонд издает газету для школьников «Детский компьютерный мир». Издается журнал «Электрон и компьютер», в котором имеется рубрика «Друзья школьников — компьютеры». Выходит специальная литература. Например, изданы книги: «Как использовать Бейсик», «Как использовать Лого», «Микрокомпьютер — друг детей», «Как описывать Бейсик и Лого», «Интересный язык — Бейсик». По телевидению организованы передачи под рубриками «Цикл лекций по Бейсику», «Компьютерное образование», «Телерепортаж с компьютерных соревнований».

Семинар «Проблемы компьютеризации дошкольного воспитания»

Несколько лет НИИ дошкольного воспитания АПН СССР и ЦНИИ «Электроника» МЭП СССР ведут совместную глубокую научно-практическую разработку проблем компьютеризации дошкольного уровня образования. В рамках этого сотрудничества успешно развивается новая информационная технология воспитания, направленная на формирование творческой, всесторонне развитой и эмоционально благополучной личности ребенка. Был создан и компьютерно-игровой комплекс для дошкольников — КИК «Электроник». Игры и занятия на базе КИК «Электроник» обеспечивают развитие у детей общих умственных способностей, инициативности, стремления к сотрудничеству с воспитателями и другими детьми, позволяют детям не только усваивать новые знания, но и результативно применять их как в самостоятельной деятельности, так и в коллективной.

Госкомитет СССР по народному образованию утвердил подготовленную НИИ ДВ АПН СССР совместно с ЦНИИ «Электроника» комплексную межотраслевую целевую исследовательскую программу «Компьютерно-игровой комплекс в системе дошкольного воспитания». В планах программы — разработка концепции компьютеризации дошкольного воспитания, а также санитарно-гигиенических, эргономических, дизайнерских требований к организации и условиям использования КИК в дошкольных учреждениях; создание систем развивающих дидактических игрушек, пособий и других материалов, необходимых для творческого развития личности ребенка в условиях применения компьютеров для овладения компьютерной грамотой; разработка развивающих компьютерных игр для дошкольников; подготовка педагогических кадров для работы с КИК, проработка возможностей использования КИК в новых направлениях.

Первые данные об использовании компьютерной технологии обучения дошкольников свидетельствуют о благоприятной (с учетом создания необходимых условий и режима работы) эмоционально-личностной реакции у детей, общем развивающем и совершенствующем влиянии компьютерной технологии на весь педагогический процесс детского сада, свидетельствуют о наличии пока еще малоизученных резервов совершенствования всей системы воспитания, развития и обучения дошкольников.

В апреле 1989 г. в Москве был проведен первый научно-практический семинар по проблеме применения компьютеров в системе дошкольного воспитания. В задачу семинара входило обсуждение с широким кругом специалистов первого опыта и перспективных направлений использования компьютерной техники в дошкольных учреждениях.

В работе семинара приняли участие более 300 человек, среди которых были представители практически всех республик страны. В числе участников семинара — заведующие и методисты, воспитатели дошкольных учреждений, преподаватели вузов и педагогических училищ, методисты и инспекторы по дошкольному воспитанию отделов народного образования. На семинар были приглашены ведущие ученые и специалисты страны в области дошкольного воспитания и смежных отраслях знаний: медики, эргономисты, физиологи, дизайнеры. Присутствовали на семинаре и представители промышленных предприятий, которые имеют дошкольные учреждения (некоторые из них уже стали обладателями КИК «Электроник», многие рассчитывают получить их в текущем или будущем году).

Остановимся на проблемах, обсуждавшихся в рамках семинара.

Были заслушаны теоретические доклады об информатизации образования в целом и его первой ступени — общественного дошкольного воспитания. Концепцию создания технологии развития детей дошкольного возраста в условиях применения компьютерной техники изложил директор ЦНИИ «Электроника», д-р техн. наук, профессор Ю. Б. Митюшин. С позиций научных прогнозов совершенствования дошкольного воспитания раскрыты проблемы формирования интеллектуального развития личности дошкольников директором НИИ ДВ АПН СССР, д-ром психол. наук профессором Н. Н. Поддьяковым. Психологические аспекты компьютеризации образования осветил в своем выступлении профессор МГУ д-р психол. наук О. К. Тихомиров.

В выступлении зав. лабораторией НИИ ДВ АПН

СССР канд. психол. наук С. Л. Новоселовой раскрыты значение и особенности введения новой информационной технологии в систему дидактических средств детского сада. По ее мнению, идея воспитания поколения, имеющего с ранних лет сформированную психологическую (личностно-мотивационную, интеллектуальную, операциональную) готовность к использованию персональных и профессиональных компьютеров, компьютерных программ различного содержания и уровней, определена перспективой развития промышленной, гуманитарной, научной, экологической, бытовой сфер общества, всей его культуры. Особое внимание обращено на необходимость создания компьютерно-игрового комплекса как системы условий, позволяющих сочетать игры и занятия на компьютерах с разнообразными развивающими играми и занятиями. В этой связи компьютерные игры не заменяют обычные игры, а дополняют их, входят в их структуры, обогащая и развивая весь педагогический процесс в целом.

Ведущей темой в докладах была забота о детях. Первая заповедь, определяющее требование при введении компьютерной технологии воспитания дошкольников — «не навреди!». С этих позиций прозвучали доклады зав. лабораторией НИИ физиологии детей и подростков АПН СССР, д-ра мед. наук Л. А. Леоновой, зав. лабораторией эргономики НПО «Горсистемотехника» г. Киева канд. психол. наук В. М. Бондаровской.

Особенности использования компьютерно-игровых программ в решении задач формирования у детей культуры познания, деятельнопрактического отношения к миру отражены в докладах зам. директора НИИ ДВ АПН СССР, канд. пед. наук Л. А. Парамоновой и преподавателя Клайпедского педагогического института И. Ю. Пашелите. О проблеме выявления и формирования с раннего возраста творческой интеллектуально активной личности говорила в своем выступлении зам. председателя Московского общества психологов, ст. науч. сотр. НИИ общей и педагогической психологии АПН СССР Д. Б. Богоявленская.

Особо выделены вопросы использования компьютерно-игровых программ в структуре сюжетно-ролевых игр, методики работы с детьми в компьютерно-игровом комплексе (доклады ст. науч. сотр. НИИ ДВ АПН СССР Е. В. Зворыгиной, ст. преподавателя Бельцкого педагогического института Г. П. Петку, ст. преподавателя Горьковского педагогического института канд. пед. наук Н. Ф. Комаровой). Заинтересовал присутствующих опыт работы с детьми в компьютерно-игровом комплексе, который был обобщен в докладах зав. детским садом П. Д. Маханевой (г. Горький) и зав. детским садом Т. И. Кайряк (г. Бельцы).

Профессор Шяуляйского пединститута, д-р пед. наук Б. И. Бальчитис рассказал о своей работе в компьютерном классе с шестилетками и детьми начальных классов.

О реализации и перспективах совместных работ с другими научно-исследовательскими институтами, предприятиями и организациями по развитию КИК и о перспективах разработки технических и программных средств КИК сообщил участникам семинара начальник лаборатории ЦНИИ «Электроника» Ю. М. Горвиц. Проблемам и методам организации производства и поставок КИК, абонентного обслуживания заказчиков, а также другим вопросам, связанным с маркетингом КИК, в том числе на зарубежном рынке, посвятил свое выступление начальник отделения ЦНИИ «Электроника» В. Ю. Демьяненко.

Перед участниками семинара выступил заместитель председателя Гособразования СССР Ф. И. Перегудов. Его доклад был выслушан с большим вниманием, после чего участники семинара обратились к Ф. И. Перегудову с рядом вопросов и критических замечаний, с практическими предложениями. В числе важнейших выступавшие назвали проблему предоставления статуса экспериментальных (с соответствующими правами) тем дошкольным учреждениям, которые по своей инициативе внедряют передовые методы воспитания, в том числе и новые информационные технологии.

В дни работы семинара его участники смогли ознакомиться с экспериментальной работой компьютерно-игрового комплекса в яслях-саду № 1623 «Электроник» Гагаринского района Москвы, где были продемонстрированы развивающие компьютерные программы для детей 4—6 лет, оборудование игрового и физкультурного зала, некоторые педагогические приемы организации занятий детей с КИК.

По мнению участников семинара, процесс внедрения компьютерной технологии в систему дошкольного воспитания должен проходить постепенно, по мере создания необходимой материальной базы, достижения каждым дошкольным учреждением определенного уровня воспитательно-образовательной работы с детьми, соответствующей подготовки кадров, при строгом соблюдении всех необходимых психолого-педагогических, санитарно-гигиенических и эргономических требований. При этом следует продолжить глубокую всестороннюю научно-практическую проработку указанных проблем; обеспечить дошкольные учреждения необходимыми программно-методическими материалами, научно обоснованными требованиями и рекомендациями по созданию компьютерно-игровых комплексов; комплектами развивающих, обучающих и диагностических программ для детей дошкольного возраста; своевременно решать вопросы материально-технического обеспечения комплексов, их абонентного обслуживания.

Участники отметили хорошую организацию семинара специалистами ЦНИИ «Электроника» и выразили пожелание сделать такие встречи регулярными, а средствами массовой информации шире освещать проблемы компьютеризации всех уровней образования, включая дошкольное.

Ю. ГОРВИЦ, С. НОВОСЕЛОВА, Н. РЕУЦКАЯ

Школа творчества

Как реликвии хранятся в некоторых школах Ленинграда грамоты и дипломы конференций, подписанные академиком А. П. Ершовым, вызывая восхищенные вопросы ребят, приходящих поступать в эти школы...

Ленинградские научно-практические конференции «Школьная информатика» уже хорошо известны в стране. Проходят они ежегодно с 1981 г.; 10—11 марта 1989 г. состоялась восьмая.

От 52 до 240 докладов, от 130 до 500 участников, от 3 до 10 секций — таков количественный «диапазон» конференций; от пионеров до академиков — ее «кадровый диапазон».

Именно на этих форумах десятки учителей, сотни школьников впервые услышали о персональных компьютерах от Андрея Петровича Ершова, выступления которого оказали колоссальное воздействие на аудиторию: вселили оптимизм, дали надежду, зарядили пониманием проблем, активностью. А время, начало 80-х, было совсем другим.

Первые конференции часто наталкивались на непонимание. Легко, свидетельствую это как учитель физико-математической школы, было лишь с учениками: они всегда верили, что ЭВМ в школе нужна и, значит, рано или поздно будет.

Как проходила подготовка к конференциям в школе?

После выявления желающих обдумывались и конструировались выступления: многие ребята имели «за душой» опыт не только школьной программистской деятельности (занимались в кружках при вузах и т. п.). Формулировали темы, писали тезисы, выясняли заодно, что школьники, за редким исключением, совершенно не умеют ни того, ни другого. Приходилось действовать методом «последовательных приближений»... Стали репетировать доклады — выяснилось, что выступать большинство старшеклассников тоже не в состоянии. Нет, конечно, рассказать у доски о выводе формулы, доказательстве теоремы, экономическом положении Нигерии они могут, но сформулировать задачу, показать проблемы и наметить пути их решения и не просто перечислить операторы программы, а пояснить нюансы, оказалось, не умеет почти никто. Пришлось отрабатывать с учениками эти навыки, проявлять терпение, стараться не обидеть. Зато сколько было у ребят радости, гордости, удовлетворения от хорошо выполненного дела, когда они, совершенно по-взрослому держась на докладе, солидно отвечали на вопросы председателя секции и слушателей, отстаивали свою точку зрения! А много ли в сегодняшней школе у учащихся таких возможностей? Согласитесь, мало.

Помню, как поразил всех на первой конференции уровень докладов новосибирских восьмиклассников. Один преподаватель ленинградского вуза говорил: «Если бы наши студенты так защищали курсовые!» Как приятно было видеть на последней конференции одну из тогдашних «возмутительниц спокойствия» уже в ранге научного сотрудника!

Еще одна картинка с конференции: после доклада десятикласснику (он получил диплом I степени) четыре профессора задавали вопросы (и совсем не из вежливости) более получаса. Безусловно, подобные конференции — это подлинная школа воспитания, научного и творческого становления. Не случайно многие школьники, прошедшие эту школу, с первых же курсов вузов включаются в научно-исследовательскую работу, хорошо «вписываются» в эксперименты кафедр, имеют самостоятельные научные интересы, серьезные заделы.

А у организаторов конференции — работников Комитета по народному образованию исполкома Ленсовета, преподавателей Ленинградского института авиаприборостроения, других вузов, учителей, работников научных учреждений — свои заботы: как заранее спланировать набор секционных докладов, как разместить иногородних участников, как максимально объективно (ведь перед нами дети!) оценить доклады, как провести предварительный отбор докладов (заявок очень много).

По просьбе участников восьмой конференции оргкомитетом принято решение начиная с девятой называть эти конференции именем А. П. Ершова.

Б. МЕДВЕДЬ,
член оргкомитета

●

Организаторы конференции — истинные подвижники и энтузиасты — взвалили на себя огромный труд и ответственность: прослушать и справедливо оценить работы учащихся. VIII конференция проводилась в хорошо приспособленных для этого аудиториях Дворца пионеров и школьников и Дома технической пропаганды. Мы благодарны ленинградцам за организацию и проведение конференции, приглашение и высокую оценку наших ребят.

Грамоты за I место получили, например, такие работы:

«Анализ и расчет обобщенных сетевых моделей» — программа предназначена для составления простого календарного плана и сжатого календарного плана на основе известных технологических связей между отдельными работами и директивных ограничений. Программа функционирует на основе базы данных. Работа с программой ведется с помощью иерархически построенных меню в диалоговом режиме. Она может использоваться в строительных трестах и при проведении научных исследований в институтах.

«Моделирование обдувания бесконечного цилиндра с произвольным сечением в аэродинамической трубе и поведение бесконечного цилиндра в однородном магнитном поле».

Грамотами за II место, в частности, были отмечены работы:

«Разработка алгоритмической структуры экспертной системы и программная реализация ее отдельных компонент» — программы, управляющие экраном при работе эксперта с системой принятия решений, а также программа, позволяющая вести диалог при формировании вопроса.

Программа используется в системе принятия управленческих решений с тестированием (СПУРТ).

«Расчет нестационарного температурного поля в плоской тепловыделяющей пластине» — программа рассчитывает температуру в каждой точке плоской бесконечной пластины. В результате решения задачи получаем картину распределения температур в каждый момент времени в пластине.

Из дружеских побуждений мы хотели бы отметить то, что нам понравилось, и то, что, на наш взгляд, можно совершенствовать в дальнейшем.

Мнения учащихся:

на конференции удалось сравнить свою работу с работами других школьников, это было интересно;

не ожидала, что буду так волноваться; жаль, что не услышала ни замечаний, ни вопросов, ни реплик, хотя были другие доклады, которые вызвали реакцию зала и жюри;

получил большое удовольствие от своего выступления, заинтересованного внимания слушателей;

в секции «Обучающие программы», к сожалению, не было возможности показать работу обучающих программ на компьютере;

жюри и слушатели мало задавали вопросов;

доклады некоторых конкурентов производили впечатление слабых работ, не подкрепленных распечатками программ и результатов;

организаторы конференции не требовали рефератов докладов для предварительного отбора участников;

на пленарных заседаниях мало рассказывали интересного для ребят;

Мнения учителей:

для каждого школьника выступление на конференции — первый в жизни публичный отчет о проделанной работе, поэтому педагогам нужно уделять внимание подготовке таких выступлений (как построить сообщение, расставить акценты, рассказать о преодоленных трудностях и найденных новациях и в то же время уложиться в регламент). От этой работы зависит уровень выступлений на секциях и конференции в целом;

предварительный отбор участников недостаточно проводить по краткой аннотации и названию доклада;

уровень проделанной учащимися работы жюри могло бы оценивать как по докладу, так и по распечаткам программ и результатов или по демонстрации работы программ;

так называемые пленарные заседания нельзя строить по аналогии конференции для взрослых: на этих заседаниях хотелось бы знакомить участников конференции с ведущими специалистами в области информатики, последними достижениями в этой области, зарубежным опытом, но в интересной для школьников форме (!);

положительный воспитательный эффект можно ожидать от отсутствовавшей на данной конференции культурной программы, которая позволила бы познакомить участников конференции друг с другом в непринужденной обстановке после завершения состязания; обмен мнениями, информацией, программами, адресами способствовал бы развитию школьной информатики в нашей стране.

До встречи, IX Ленинградская конференция, в 1990 г.!

Н. ПАХОМОВА,
учитель СШ № 444 Москва

Информатика в педучилище

Введение предмета «Основы информатики и вычислительной техники» в учебный план педагогических училищ открывает возможность подготовки кадров народного образования для начальной школы и дошкольных учреждений на более высоком уровне. Но механически введенная в педучилище программа школьного курса ОИВТ не содержит главной компоненты: ориентации на профессиональную подготовку учителей начальной школы и воспитателей дошкольных учреждений. Преподавателям информатики в педучилищах предстоит решить важную проблему: как построить курс ОИВТ, чтобы он преследовал не только общеобразовательные цели, но и профессиональные, т. е. остро встал вопрос о выработке единой политики по внедрению информатики и вычислительной техники в педагогических училищах республики. Уже сегодня 150 училищ имеют компьютеры и оборудованные классы, и каждое педагогическое училище ищет свой подход, разрабатывает свои программы по активизации и совершенствованию учебного процесса. Эта тенденция объясняется тем, что многие преподаватели ОИВТ пришли в педучилище, не имея педагогической подготовки.

Для рассмотрения этих вопросов в декабре 1988 г. в Московском педагогическом училище № 1 им. К. Д. Ушинского с участием Министерства просвещения РСФСР было проведено совещание-семинар преподавателей ОИВТ педагогических училищ. На этом совещании была создана учебно-методическая комиссия, перед которой министерство поставило задачу объединить усилия и четко определить, какими проблемами занимается каждый регион нашей республики. Комиссия ставит перед собой следующие цели: разработать содержание предмета, координировать преподавание ОИВТ в педагогических училищах, оказывать методическую помощь преподавателям ОИВТ, изучать и обобщать накопленный опыт использования ВТ в учебно-воспитательном процессе в СССР и за рубежом, способствовать компьютеризации народного образования.

В Министерстве народного образования РСФСР на базе МПУ № 1 им. К. Д. Ушинского в июне 1989 г. состоялся очередной семинар этой учебно-методической комиссии.

В семинаре приняли участие: члены комиссии, преподаватели Москвы, сотрудники ИПС АН СССР, ЦНИИ «Электроника», НИИ дошкольного воспитания АПН СССР и члены научно-учебного комплекса компьютеризации образования, создан-

ного на базе МОПИ им. Н. К. Крупской.

Главной темой было обсуждение роли и места курса ОИВТ в повышении качества подготовки будущих учителей начальной школы, воспитателей дошкольных учреждений. Отмечалось, что сегодня вычислительная техника активно внедряется как в учебный, так и в воспитательный процесс, в организацию свободного времени детей и школьников.

Было отмечено, что еще не все преподаватели да и руководители учебных заведений с должным пониманием и ответственностью относятся к вопросу внедрения вычислительной техники в учебный процесс. Крайне слабо используется техника для проведения диагностирования, разработки и составления психолого-педагогических задач, новых сюжетно-ролевых игр с применением электронных игрушек. Особенно мало внимания уделяется совместной работе преподавателей курса ОИВТ и преподавателей частных методик.

Педагогические училища готовят специалистов — педагогов и воспитателей. Встает вопрос о целесообразности копирования сокращенного курса школьной информатики. Действительно, время, отпущенное на курс ОИВТ, в педучилище урезано наполовину, материал о вычислительной технике «разбросан» по двум различным дисциплинам, контингент учащихся — в основном девушки с гуманитарным складом мышления, и, наконец, необходимо акцентировать внимание на профессиональной компоненте.

Последнее означает, что курс информатики должен быть нацелен на решение профессиональных задач, таких, как умение пользоваться пакетом общих программ для профессиональных целей. Одновременно выпускники училища должны ориентироваться в обучающих программах и знать основные алгоритмы их исполнения, оценивать воспитательное и развивающее воздействие программ на детей и т. п.

С этой точки зрения семинар вызвал несомненный интерес. На нем были представлены два программно-методических комплекса для детей младшего возраста. Это «КИК» — компьютерный игровой комплекс для детей дошкольного возраста, разработанный в ЦНИИ «Электроника» и НИИ дошкольного воспитания АПН СССР, и «Роботландия», разработанный в ИПС АН СССР для младших школьников. Оба пакета достигли уровня товарного продукта и готовы для использования в дошкольных учреждениях и школах. Педагогический эффект пакетов достоин самого пристального изучения.

Участники семинара ознакомились с подходами к разработке учебников и пособий для школьных и дошкольных отделений педагогических училищ на базе этих комплексов. Можно спорить о подходах, о целях компьютеризации учебно-воспитательного процесса дошкольников и школьников, однако нельзя не приветствовать появление первых программ такой ориентации. Знакомство преподавателей ОИВТ педагогических училищ с представленными работами было полезным и конструктивным. При составлении учебных программ по ОИВТ целесообразно учитывать имеющиеся комплексы.

Преподавателями 1-го и 12-го педучилищ Москвы были представлены программы обучающего характера, разработанные силами учащихся на уроках информатики. По общему мнению семинара, эта работа признана полезной и заслуживающей поддержки.

С 1 сентября 1989 г. в педагогических училищах вводятся новые учебные планы, при реализации которых большое внимание должно уделяться развитию дальнейшей самостоятельности педагогических коллективов в выборе путей подготовки педагогических кадров на новом качественном уровне. В соответствии с этими планами 20 % учебного времени отводится в непосредственное распоряжение учебных заведений; они могут проявлять инициативу, вносить изменения и дополнения в зависимости от местных условий и состояния материально-технической базы. Существенная часть этого времени должна быть направлена на реализацию возможностей ВТ в каждой учебной дисциплине.

С. РАТКИН,
преподаватель ОИВТ Норильского педагогического училища
Е. ОВЧИННИКОВ,
преподаватель ОИВТ МПУ № 1 им. К. Д. Ушинского
А. СМЯТСКИХ,
начальник отдела МНО РСФСР

Соревнуются техникумы

В мае 1989 г. в Запорожском техникуме электронных приборов состоялась олимпиада по ОИВТ среди техникумов отрасли. Администрация техникума подготовила прекрасную базу для проведения олимпиады, была предусмотрена обширная культурная программа для участников.

В целях обмена опытом проведения такой предметной олимпиады уместно обратиться к конкурсным задачам и самой процедуре проведения олимпиады. Участникам было предложено за 2,5 ч решить четыре задачи. Каждое решение — программа на Бейсике с выполнением контрольного просчета. Оценивались задачи разным количеством баллов, исходя из заранее определенного уровня сложности. Главными критериями оценки были работоспособность и оптимальность программ, а также оригинальность мышления, что поощрялось дополнительным баллом.

Авторство работ было зашифровано. Каждому участнику присваивался шифр, а список соревнующихся и соответствующий им шифр хранились у одного из членов жюри, который не занимался проверкой решений задач. Только после подведения итогов по шифрам были определены и фамилии участников.

Участникам олимпиады были предоставлены все возможности для творческой работы. Каждому

была выделена одна ЭВМ, не было ограничений в использовании различных версий Бейсика.

Проверка решений задач проводилась членами жюри непосредственно на компьютере, после ввода программы с фиксацией времени. Таким образом проверялась работоспособность программы.

Дипломы за I командное место в олимпиаде по ОИ и ВТ среди техникумов отрасли были присуждены Винницкому ТЭП, 36 баллов; за II командное место — Запорожскому ТЭП, 24 балла; за III командное место — Саратовскому ТЭП, 21 балл.

Почетными грамотами в личном первенстве награждены учащиеся ССУЗов отрасли:

А. Усатюк — I место, Винница, 21 балл;
Р. Рутман — II место, Винница, 15 баллов;
А. Жмыхов — II место, Запорожье, 13 баллов;
И. Попков — III место, Москва, 12 баллов;
А. Романчук — III место, Саратов, 11 баллов;
В. Жук — III место, Запорожье, 11 баллов.

Ограниченное количество времени не позволило многим учащимся справиться с решением всех задач. Общим недостатком явилось усложнение и запутанность решений, выбор неоптимальных путей.

Претензий от участников олимпиады и представителей команд к членам жюри не поступало. Во всех вопросах проведения олимпиады была проявлена объективность, чему способствовало разделение функций членов жюри: каждый отвечал за определенный участок. Кроме того, окончательное решение в выборе конкурсных задач было принято коллегиально, непосредственно за 30 мин до начала олимпиады. В целом уровень сложности задач был достаточно высоким.

Помещаем несколько конкурсных задач. (Решения будут опубликованы в следующем номере.)

Задача № 1

Условие:

Составить программу нахождения суммы цифр заданного натурального числа n. Найти Σ цифр для n=3841; n=45238. *Оценка* — 4 балла.

Задача № 2

Условие:

Составить программу, которая определяла бы временной интервал в днях между двумя датами.

Найти с ее помощью, сколько дней прошло со дня первого полета человека в космос до настоящего времени.

Оценка — 7 баллов.

Задача № 3

Условие:

Определить норму заданной матрицы, т. е. число

$$\max \left(\sum_{j=1}^{n} |a_{ij}| \right); \ 1 \leqslant i \leqslant m, \ j = 1.$$

Решить для:

2	—3	5	7
—11	26	—39	8
9	13	17	—21

Оценка — 5 баллов.

Задача № 4

Условие:

Найти такую точку заданного на плоскости множества точек, сумма расстояний от которой до остальных минимальна.

Решить для: A(2,1), B(1,3), C(2,4) D(4,3), E(5,5).

Оценка — 6 баллов.

А. БЛИННИКОВ,
И. КИНДРАСЬ,
г. Запорожье

Интеринформ

Так называлась проходившая в июне в Москве выставка новых информационных технологий. Ее организовали Международный центр научной и технической информации и австрийская фирма «Квадрифолио В. у. И.», чтобы помочь австрийским бизнесменам выйти на советский рынок.

Впрочем, некоторые с этой задачей справились самостоятельно — «Зема» продает свои компьютеры (PC-совместимые), локальные сети, периферийные устройства через совместное предприятие «Союзфоринвест», а «Тродат» торгует своими штемпелями на ул. Горького.

Вы можете спросить: как штемпели оказались среди новых информационных технологий? Очень просто. Во-первых, это штемпели новые: удобные, красивые, самосмачивающиеся краской; владелец может набрать из прилагаемых резиновых букв любой текст. Во-вторых, штемпель — хотя и не очень заметный, но весьма полезный компонент конторского быта, что остро ощущала редакция «ИНФО», пока три года стояла в очереди на изготовление этой нехитрой вещи в московской мастерской.

«СТМ» — фирма торговая, представляет интересы производителей из Австрии, ФРГ, Японии, отсюда — разнообразие ассортимента: телефаксы, принтеры, компьютеры, клавиатуры... В частности, последнее достижение техники и дизайна — клавиатура с жидкокристаллической индикацией: каждая клавиша — небольшой экранчик. Вот на всех клавишах — латинские буквы; нажатие кнопки — клавиатура переключилась на кириллицу; еще одно переключение — возникает арабская вязь... Увы, цена подстать удобству: более 2000 долларов. Но ведь и собственное ОЗУ клавиатуры — 256К байт!

Одно из отделений фирмы «Филипс» показывает диктофоны (в том числе и кар-

манные) системы «Мини». Похожие всем нам неоднократно приходилось видеть в заграничных фильмах, но у этих есть особенность: крайняя простота конструкции и — как следствие — высокая надежность. Разобранный диктофон ошеломляет: моторчик, две шестеренки, блок головок, переключатель, печатная плата с микросхемой и дюжиной резисторов — и это всё. Однако достаточно для записи в течение часа на кассету размером в половину спичечного коробка.

Телефоны фирмы «Капш» — цифровые (передают звук не в виде аналоговых электрических сигналов, а закодированным в цифровой форме, что обеспечивает высокое качество связи), но могут работать при подключении к обычным аналоговым телефонным каналам. Модели разные — от обычных с виду до телефонного компьютера с дисплеем, клавиатурой, ОЗУ на 256К байт. «Беспроводной» телефон: УКВ-передатчики и приемники в аппарате и трубке, так что с трубкой можно свободно разгуливать по комнате и даже по зданию (фирма уверяет, что на расстоянии менее 300 м и железобетонные стены не помеха).

Ну и, наконец, для любителей мастерить своими руками: «Австроплан» поставляет оборудование для производства пленочных и кнопочных клавиатур. Производственное помещение площадью 1500 м^2, 22 работника, 180 кВт электропитания, 0,3 м^3/ч воды, 16 м^3/ч сжатого воздуха — и можно выпускать в год 50 тыс. пленочных клавиатур для ЭВМ или 200 тыс.— для телефонов. Фирма обеспечивает поставку и монтаж оборудования, обучение персонала, передачу «ноу-хау» за 2,8 млн. долларов. Для налаживания производства 120 тыс. IBM-совместимых кнопочных клавиатур в год нужно 4 млн. долларов. Возможны компенсационные сделки.

Желающие есть?

Офицеры обсуждают проблемы компьютеризации

Остро стоят перед училищами и академиями проблемы компьютеризации, задачу овладения всеми выпускниками военных учебных заведений навыками работы с ЭВМ поставил Министр обороны. Путям решения этой задачи был посвящен межвузовский семинар по внедрению электронной вычислительной техники в учебный процесс военно-учебных заведений. Семинар проходил в феврале 1989 г. на базе Серпуховского высшего военного командно-инженерного училища ракетных войск им. Ленинского комсомола.

Выбор места проведения семинара не случаен. Училище имеет неплохую учебно-лабораторную базу. Участники семинара могли убедиться в этом, ознакомившись с использованием компьютеров на лабораторных работах и в лекционных аудиториях, при изучении иностранных языков и специальных дисциплин, в управлении учебным процессом и при организации досуга курсантов.

Обращаясь к участникам, руководитель семинара профессор В. Н. Умников отметил, что сейчас не стоит вопрос о том, должны ли быть компьютеризированы военные училища. Вопрос заключается в том, каким путем двигаться к этому. Самое главное — не ждать руководящих указаний, новых штатов и инструкций, а активно внедрять имеющуюся технику, по прямым связям с предприятиями в рамках совместных научно-исследовательских работ получать современные ЭВМ.

Проблемы компьютерного обеспечения деятельности военно-учебных заведений осветил начальник училища И. И. Можаев, уделив особое внимание тому, что уже сделано в училище, каким путем идет компьютеризация учебного процесса.

Большой интерес участников семинара вызвало обсуждение вопросов разработки и внедрения автоматизированных обучающих систем. В. И. Лукьянов и И. П. Солин, дав анализ существующих зарубежных и отечественных АОС, рассказали о собственной разработке — АОС «Эрудит», работающий на базе мини-ЭВМ. Разработчики ожидают снижение времени подготовки 1 ч компьютеризированных занятий с 40—120 ч в существующих АОС до 10—20 ч.

Ю. Ф. Кривой рассказал об использовании АОС типа «Ирис» при подготовке офицеров-политработников, где основной упор пока делается на автоматизированный контроль знаний. Н. С. Корнеев изложил оригинальный подход к формированию курсантских групп на основе социально-психологических исследований с помощью ЭВМ.

Принципиальным вопросам компьютеризации учебного процесса и формированию компьютерной грамотности курсантов были посвящены доклады С. А. Полянского и О. А. Козлова. Отмечая необходимость непрерывной компьютерной подготовки, докладчики подробно остановились на проблемах ее внедрения. Подчеркивалась необходимость использования учебно-ориентированных программных средств, а также ориентация на персональные компьютеры, в частности на КУВТ-86 и «Электроника БК-0010».

Рассматривая проблемы и задачи компьютерной подготовки, В. В. Илющенко отметил существенное повышение интереса курсантов к изучению программирования за счет широкого ис-

пользования КУВТ-86. Кафедра сделала эти компьютеры доступными для каждого курсанта, обеспечив двухсменную работу четырех классов, в результате в два раза уменьшилось количество неудовлетворительных оценок, на 20—25 % увеличилось количество хороших и отличных оценок, руками курсантов созданы сотни программ для учебного процесса. Отмечалась также достаточно высокая надежность БК-0010 при весьма интенсивной эксплуатации.

Различным прикладным вопросам внедрения ЭВМ на лабораторных и практических занятиях, организации эксплуатации ЭВМ были посвящены выступления В. Л. Котова, В. И. Фрункера и А. С. Омельченко, В. В. Евтушенко и других.

Среди стендовых докладов следует отметить работу Ю. Ф. Михайлова об использовании компьютера для чтения лекций. В училище оборудовано несколько аудиторий, в которых установлены большие цветные телевизоры, связанные с вмонтированной в пульт преподавателя «Электроникой БК-0010». При этом у преподавателя появляются новые возможности для подготовки компьютерных лекций.

В. В. Храмов говорил о количественных формах оценки эффективности использования компьютеров в учебном процессе и продемонстрировал это на конкретных примерах по результатам сдачи экзаменационной сессии.

Работа семинара не ограничивалась одними докладами и сообщениями. Гостеприимные хозяева предоставили возможность всем участникам ознакомиться с практическим решением вопросов компьютеризации в училище. Такое обилие средств вычислительной техники и программного обеспечения многие видели впервые. Широко используются КУВТ-86 и отдельные машины «Электроника БК-0010». Они есть практически на каждой кафедре, постоянно загружены. Многие кафедры оборудуют вместе два КУВТ-86, что позволяет организовать 24 рабочих места и обеспечить высокую надежность проведения практических и лабораторных работ — машины ДВК-3 могут «подстраховать» одна другую.

Специальные кафедры показали оригинальные комплекты: машина типа СМ-4 + БК-0010 + штатная аппаратура. Такие комплексы позволяют проводить практические и лабораторные занятия с элементами штатной техники, моделируя сложные процессы с помощью мини-ЭВМ, демонстрируя при этом схемы, переходные процессы, панели аппаратуры на графических дисплеях БК-0010. Этот вариант использования компьютеров привлек внимание многих участников семинара.

Как и другие военно-учебные заведения, училище располагает мощным парком ЕС ЭВМ, дисплейные классы которых есть уже на ряде кафедр. Можно было ознакомиться и с организацией работы, и с методикой проведения занятий, и с программным обеспечением. Активно ведутся в училище работы по внедрению баз данных.

Но не только на занятиях используются компьютеры в Серпуховском ВВКИУ. Есть в подразделении одного из факультетов видеокомпьютерный центр, сделанный руками курсантов. Теперь они могут в свободное время посмотреть

вчерашнюю программу «Взгляд», поработать за БК-0010 или просто поиграть.

Особо хотелось бы отметить следующее. Подобные семинары, конференции, выставки последнее время проходят интересно до тех пор, пока выступающему не задается вопрос о возможности получения разработанных программ. Тут начинаются разговоры о трудоемкости этой работы, о кооперативах, но переписывать программы никто просто так не дает. Хозяева семинара охотно делились всем, поэтому на второй день работы участники разошлись по кафедрам, где переписывали на ленты и дискеты интересующие их программы, перерисовывали схемы подключения и фотографировали плакаты, делали выписки из методических разработок и обменивались ими. Эта сторона работы семинара понравилась всем участникам.

Участники семинара отмечали большую эффективность использования в учебном процессе как КУВТ-86, так и отдельных машин «Электроника БК-0010». Применение этих компьютеров позволило действительно начать компьютеризацию учебного процесса, вовлечь в круг проблем компьютеризации многих преподавателей и курсантов. Поэтому многие участники семинара были удивлены п. 2 «Рекомендаций IV Всесоюзного семинара «Разработка и применение программных

средств ПЭВМ в учебном процессе» (см. Информатика и образование. 1989. № 1. С. 126), в котором призывается прекратить поставку этого класса компьютеров в учебные заведения. Скорее можно было услышать сетования на нехватку так гармонично вписавшихся в учебный процесс дешевых и доступных персональных компьютеров типа «Электроника БК-0010».

В решении, принятом на семинаре, отмечалась необходимость улучшения снабжения военных училищ вычислительной техникой и особенно персональными компьютерами, а также формирования пакетов (банков) учебных, обучающих программ, ведения в специальных каталогах учета учебных программ на все виды занятий.

Хотелось бы, чтобы подобные семинары стали постоянной формой общения и обмена программным и методическим обеспечением.

О. КОЗЛОВ

Веселый урок

Бейсико-военно-полевой словарь

AUTO — продолжать самостоятельно!
BIN — на 1—2 рассчитайсь!
BEEP — ура!!!
BYE — отбой!
CLS — оправиться!
CONT — продолжать упражнения!
DATA — список ориентиров
DIMENSION — оперативный простор
EDIT — выйти из строя!
END — расчет окончен!
ERROR — разрешите доложить!
GOSUB — наряд на службу
GOTO — шагом марш!
INPUT — заряжай!
KEY — пароль
LINE — линия огня
LIST — список личного состава
LOCATE — по местам стоять!
NEW — разойдись!
NOT — никак нет!
OUT OF DATA — не могу знать!
OK — есть!
PRINT — изложить письменно!
PSET — опорный пункт (огневая точка)
RENUM — разомкнись! (по порядку номеров рассчитайсь!)
RUN — бегом марш!
STOP — на месте стой! (1—2)
TRON — докладывать о каждом шаге
TROFF — прекратить разговоры!
YES — так точно!

М. ЛАРКИН

Видеокассеты помогают программистам

Проблема нехватки программистов высокой квалификации, особенно владеющих таким сложным языком программирования реального времени, как Ада, остро стоит даже в такой компьютеризированной стране, как США.

«Где вы, хорошие Ада-программисты?» — вопрошает популярный американский журнал «Компьютер», и сам же отвечает на свой вопрос: «Вы — везде! Только надо чуточку поработать над собой». А для того чтобы работалось лучше, журнал предлагает несколько комплектов видеокассет, каждый из которых предназначен для будущих высококлассных программистов с разной начальной подготовкой. Курс языка для начинающих программистов состоит из 27 видеолент, на которых записаны лекции и семинары по введению в язык, принципам проектирования программного обеспечения и использованию конструкций языка. Хорошо подготовленным программистам, уже прошедшим практическую школу работы с языком, предлагается высококачественный Ада-компилятор, программное обеспечение для организации сети и другое сопутствующее программное обеспечение, предназначенное для обеспечения полного цикла разработки Ада-программ на персональной IBM PC AT.

После того как вы освоите все, что вам было предложено, вас с удовольствием примут на фирму, сотрудники которой и разработали программное обеспечение, пользуясь которым удалось повысить ваш профессиональный уровень.

Осталось сделать последний шаг

В сентябре 1988 г. ЭВМ («Hitech», Университет Карнеги-Меллона, США) обыграла в шахматы гроссмейстера — бывшего чемпиона США А. Деккера. Двумя месяцами позже от другой ЭВМ («Deep Thought»), созданной в том же университете, потерпел поражение Б. Ларсен.

Создатели «Deep Thought» намерены увеличить число процессоров машины,— возможно, тогда она сможет обыграть и чемпиона мира.

**Московский городской комитет
по народному образованию
Московский городской институт
усовершенствования учителей**

ПРИГЛАШАЕТ:

методистов и учителей информатики на курсы по научно-методическим основам информатики по системе:

72+36+72=180

72 часа — первоначальная подготовка по научно-методическим основам информатики и использованию ЭВМ в школе;

36 часов — умения и навыки работы на различных КУВТ (КУВТ-86, «Агат», «Корвет», УКНЦ, «Ямаха»), знакомство с программным обеспечением;

72 часа — углубленная подготовка в области информатики и программирования, знакомство с методикой углубленного преподавания курса в старших классах;

сотрудников организаций и предприятий на курсы «Компьютерная грамотность», готовящих пользователей для работы на ПЭВМ различных типов: ДВК, «Ямаха», «Корвет».

Оплата по безналичному расчету.
Справки по телефону: 151-59-31

ПРЕДЛАГАЕТ:

учебно-методический комплекс по машинному варианту преподавания курса «Основы информатики и вычислительной техники» под редакцией В. А. Каймина.
В составе комплекса:

учебник «Информатика 9—10»;
методическое пособие для учителей;
сборник задач с решениями;
пакет программ для КУВТ-86 (Бейсик, Фокал);
учебно-методический видеофильм.

Б1004 11-41

60 коп.
Индекс 70423

OldPC.su
7002
музей компьютеров

ИНФО

5 '89

ИНФОРМАТИКА
И ОБРАЗОВАНИЕ

ISSN 0234—0453. Информатика и образование, 1989, № 5, 1—128.