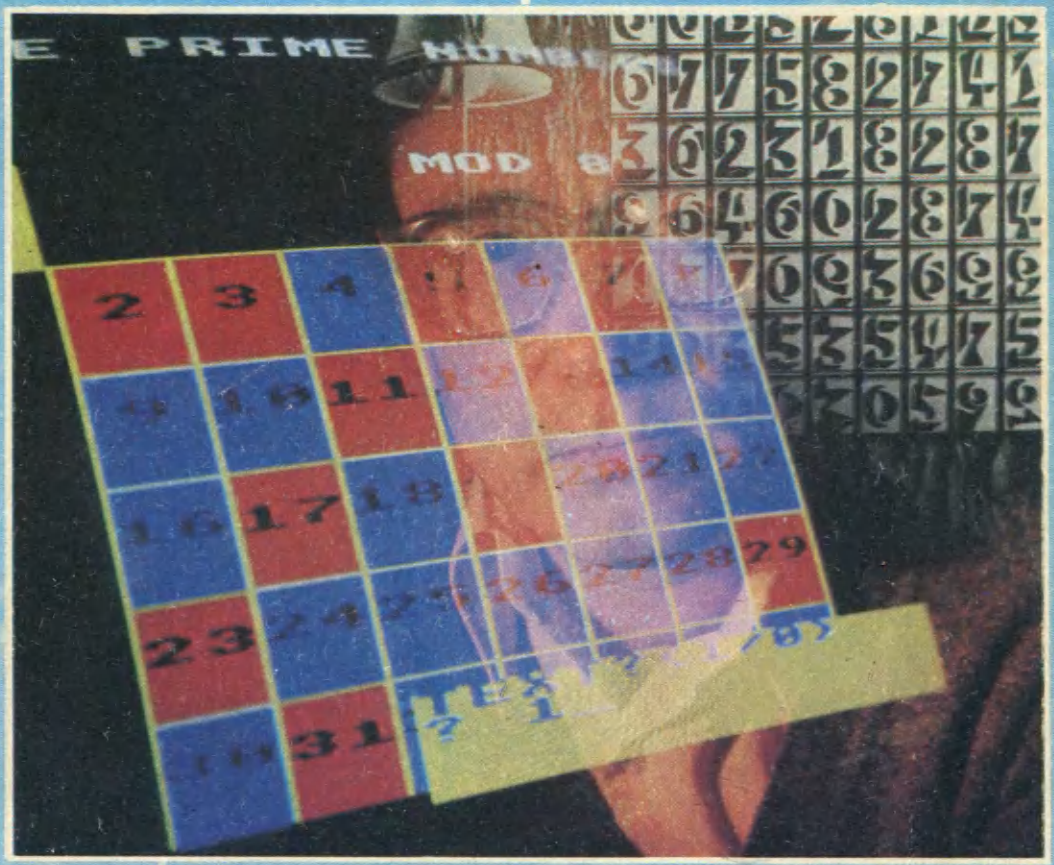
ISSN 0234—0453

ИНФО

ИНФОРМАТИКА И ОБРАЗОВАНИЕ

1 1989



Восстановить текст на праславянском или древнерусском языках поможет система «СИМВОЛ»

(статью о ней читайте в этом номере)

Научно-методический журнал
Государственного комитета СССР
по народному образованию

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

ИНФОРМАТИКА И ОБРАЗОВАНИЕ

1'89
январь — февраль

Содержание

Точка зрения

Зарубежный опыт

Нам пишут

Информация

На первой странице обложки. Магия чисел... Для одних она в мистическом восприятии «волшебных» чисел, закономерных и случайных совпадений. Для других — в радости постижения мира, открытии реальных закономерностей числового ряда.

ЭВМ — чрезвычайно полезный помощник для специалистов по теории чисел. О том, какие возможности использование машинной графики открывает для одной из древнейших ветвей математики, скоро будет рассказано в нашем журнале.

Обложка *Э. Бажилина*

В оформлении номера принимали участие *Э. Бажилин, С. Кустарева, А. Пономарев.*

Редактор отдела *А. Кравцова*
Научный редактор *Т. Драгныш*
Художественный редактор *Л. Розанова*
Зав. редакцией *Н. Игнатова*
Корректор *О. Пурлова*

Издательство «Педагогика» Академии педагогических наук СССР и Государственного комитета СССР по делам издательств, полиграфии и книжной торговли

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

Сдано в набор 22.11.88. Подписано в печать 26.12.88. А 05829. Формат 70×100/16. Печать офсетная. Усл. печ. л. 10,40. Уч.-изд. л. 12,83. Усл. кр.-отт. 42,88. Тираж 82 190 экз. Заказ 3021. Цена 60 коп.

Ордена Трудового Красного Знамени Чеховский полиграфический комбинат ВО «Союзполиграфпром» Государственного комитета СССР по делам издательств, полиграфии и книжной торговли
142300, г. Чехов Московской обл.

А. КУЗНЕЦОВ, В. ДОЛМАТОВ,
НИИ содержания и методов обучения АПН СССР

Методическая система обучения ОИВТ: структура и функции, состояние и перспективы

Как известно, методическая система обучения (МСО) по любому предмету представляет собой совокупность пяти компонентов: целей, содержания, методов, средств и организационных форм обучения данному предмету. Методические системы учебных предметов, составляющих содержание общего среднего образования, формировались на протяжении многих десятилетий в основном эмпирически, проверяясь школьной практикой и претерпевая заметные изменения с периодом порядка 10—15 лет. Для информатики же (и это видно уже сегодня) характерен динамизм в изменении ее методической системы обучения.

При анализе основных направлений развития МСО по информатике мы опирались на прогноз внедрения компьютеров и информационной технологии в среднее образование, данный в работах А. П. Ершова [1], Б. С. Гершунского [2], А. Ю. Уварова [3].

Анализ методической системы

Создание полноценной МСО по ОИВТ играет ключевую роль в ее становлении как учебного предмета. Поэтому актуальным является анализ ее компонентов, выявление наиболее узких мест и проблем, без разработки которых невозможно ее дальнейшее развитие.

Цель курса информатики — формирование компьютерной грамотности школьников, т. е. формирование совокупности знаний, умений, навыков, обеспечивающих учащимся возможность применять ВТ в учебной, а впоследствии в профессиональной деятельности.

Содержание ОИВТ включает совокупность двух взаимосвязанных компонентов: теоретического и практического. **Теоретическая** часть курса направлена на формирование у учащихся алгоритмической культуры, навыков анализа и формализации предметных задач, ознакомление с такими понятиями, как исполнитель, алгоритм, структура алгоритма, виды величин. **Практический** аспект связан с приобретением навыков работы с готовым программным обеспечением, написания программ на одном из конкретных (производственных) языков программирования. Необходимость выработки практических навыков и умений работы на ЭВМ предполагает значительное повышение удельного веса практических занятий (по сравнению с другими предметами) в общей структуре курса, придавая ОИВТ специфические черты, отличающие его от других предметов.

Слабость материально-технической базы подавляющего большинства школ, не располагавших к началу введения курса ОИВТ необходимой ВТ, привела к разрыву между теоретическим и практическим компонентами содержания учебного предмета, к деформации системы целей обучения. Это нашло

отражение и в отсутствии единства в трактовке содержания ОИВТ как общеобразовательного предмета. Многими методистами отмечались расхождения содержания излагаемого учащимся материала в разных школах и у разных преподавателей с содержанием и общей концепцией курса, отраженными в программе. Часто наблюдался уклон в сторону обучения программированию на каком-то из производственных языков.

В наиболее сложном положении оказались школы, не имевшие никакой ВТ (даже программируемых калькуляторов). Несмотря на то что сам курс ОИВТ на первом этапе его внедрения и планировался как безмашинный, реальная практика преподавания показала, что эффективность занятий с гипертрофированной (не поддержанной практикой) теоретической частью существенно снижается.

Отсутствие возможности систематического использования на уроках средств ВТ не позволяет реализовать весь общеобразовательный потенциал, заложенный в теоретической части курса, при этом значительно ослабляется и его практическая направленность, на что не раз обращали внимание как методисты, так и учителя-практики, испытывающие трудности в формировании практических навыков, предусмотренных программой. Об этом говорят и результаты массовых контрольных работ, проведенных в конце 1985/86—1986/87 учебного года НИИ СиМО АПН СССР и МП СССР.

В школах, имеющих ВТ, столкнулись с другими проблемами. Среди них следует выделить две:

разнотипность имеющейся в этих школах ВТ как по техническим возможностям, так и по программному обеспечению, используемому языку;

почти полное отсутствие ППС, специально предназначенных для программной поддержки действующего курса ОИВТ, вынуждавшее к деформации основных идей курса с креном в сторону обучения программированию на том или ином (точнее, наличном) языке программирования (начиная с Бейсика и кончая Фортраном).

Это привело к появлению разных методических систем, особенно в части используемых методов обучения и организационных форм проведения занятий. При этом основной формой введения школьников в содержание предмета оказывалась объяснительно-иллюстративная. Одной из причин этого явилось и привлечение к преподаванию инженеров-программистов, не владеющих всем арсеналом методических приемов организации занятий в условиях классно-урочной формы обучения. Перечисленные негативные моменты приводят к нарушению внутренних связей компонентов МСО.

В то же время следует отметить и положительный опыт. Практика преподавания ОИВТ выявила специфику предмета, нашедшую отражение в получивших распространение (стихийно) формах организации занятий: помимо уроков с объяснениями и решениями задач уроки-лекции, семинары, специальные практические занятия. Изменилось соотношение в пользу форм, предполагающих больший удельный вес самостоятельной работы учащихся. По существу, это отражает потребности нового предмета в более тщательной методической проработке вопроса о применяемых формах организации занятий, учитывающих его специфику.

Перспективы развития курса ОИВТ

В становлении нового учебного предмета можно выделить несколько этапов.

Первый этап (1985—1990), состоит в формировании методической системы обучения (МСО) по курсу ОИВТ, цель которого — формирование компьютерной грамотности. Изучение этого предмета в старших классах приводит к тому, что знания, полученные школьниками, не могут быть в достаточной мере использованы ими при изучении других учебных предметов. Реализация первого этапа базировалась на опыте обучения учащихся IX—X классов основам программирования на факультативных курсах и в УПК, на практике кружковой работы и летних школ юных программистов, организуемых в отдельных регионах страны.

Второй этап (1990—1995) будет связан с переносом курса в неполную среднюю школу (предположительно в V—VII классы), что позволит использовать навыки и умения, приобретенные учащимися на уроках информатики, в их учебной деятельности по другим предметам. Перенос курса потребует не только адаптации содержания предмета к возможностям школьников этого возраста, но и существенного изменения всего методического оснащения курса. Необходимым условием успешной реализации этого проекта должно быть оснащение всех школ соответствующей ВТ и программным обеспечением. На втором этапе встанет вопрос о целесообразности сохранения ОИВТ как учебного предмета в старших классах. Теоретические исследования в этой области позволяют дать положительный ответ на этот вопрос. Правда, цели и задачи обучения в старших классах при этом существенно изменятся. На первый план, в отличие от действующего курса, выдвинется уже не задача формирования компьютерной грамотности, а задача ознакомления учащихся с основами информатики как фундаментальной отрасли научного знания.

Третий этап (1995—2000) явится следующим шагом в формировании новой информационной технологии обучения, опирающейся на широкое применение средств ВТ. Этот этап, на наш взгляд, будет характеризоваться уже не просто изменением методики обучения, а коренной перестройкой всего учебного процесса. Это повлечет за собой радикальные изменения в методической системе обучения, в частности актуализирует проблему соотношения различных средств обучения: учебника и педагогических программных средств, компьютера и традиционных технических средств обучения и т. д.

Решение проблемы совершенствования МСО по ОИВТ связано с изменением всех ее компонентов:

системы целей (на основе обобщения опыта преподавания ОИВТ и теоретических исследований, проведенных за прошедший период);

содержания ОИВТ, адекватного новому пониманию целей обучения;

методов обучения, связанных с широким использованием компьютера как средства обучения (без противопоставления другим СО);

организационных форм проведения занятий, учитывающих специфику данного предмета.

Все эти изменения должны рассматриваться в тесной взаимосвязи и единстве.

Изменение в системе целей

Развитие представлений о целях курса предполагает теоретический анализ места ОИВТ в системе общего среднего образования. Это означает отказ от узкопрагматической трактовки этих целей и поиск путей решения как задач формирования компьютерной грамотности (КГ), так и задач знакомства с основами информатики как фундаментальной науки.

В противном случае, если цели курса ОИВТ будут по-прежнему связываться только с формированием КГ школьников, уже через несколько лет может встать вопрос о целесообразности изучения основ информатики в школе как самостоятельного предмета. Широкое распространение ЭВМ в школах и появление домашних компьютеров создаст ситуацию, когда формирование КГ будет осуществляться в других учебных предметах и в процессе использования компьютеров вне школы. Такую тенденцию можно наблюдать в последние годы, например, в Великобритании, где значительно сократилось число школ, в учебном плане которых есть самостоятельный учебный предмет, посвященный основам информатики.

При переосмыслении целей школьного курса информатики необходимо выделить общеобразовательные функции курса, его вклад в решение общих задач обучения, воспитания и развития школьников.

Назовем прежде всего мировоззренческую функцию, т. е. формирование научных представлений о мире. Реализация этой функции связана с раскрытием роли информационных процессов (передача, преобразование, хранение и т. д.) в живой природе, технике, обществе; значения информатики и вы-

числительной техники в развитии производительных сил, изменении характера труда человека. Надо признать, что эта функция информатики пока явно недостаточно реализована.

Общеобразовательная функция ОИВТ связана с решением задачи подготовки школьников к труду в условиях компьютеризации и электронизации всего народного хозяйства, с реализацией задач политехнического образования и профессиональной ориентации школьников. Здесь необходимо подчеркнуть роль нового учебного предмета в формировании целого ряда общенаучных умений и навыков, к числу которых следует отнести и саму компьютерную грамотность.

К числу общеобразовательных функций ОИВТ как учебного предмета следует отнести и формирование навыков использования ВТ как специфического средства решения учебных задач.

Разделение системы целей на две большие группы, связанные с формированием КГ и с введением школьников в основы информатики как фундаментальной науки, фактически приводит к выделению двух ступеней в преподавании ОИВТ. Первая связана с освоением прикладных аспектов информатики и направлена на достижение некоторого уровня КГ, обеспечивающего возможность последующего широкого использования полученных знаний и умений как при изучении теоретических основ собственно информатики, так и в других предметах. Вторая посвящена изучению основ информатики как фундаментальной отрасли научного знания и связана, прежде всего, с формированием научного мировоззрения школьников.

Отбор содержания

Изменение в системе целей приводит к пересмотру содержания учебного материала, подлежащего изучению. Можно предположить, что характерное для сегодняшнего курса тяготение к знакомству учащихся с основами программирования изживет себя, уступив место введению в основы алгоритмизации, овладению навыками грамотной постановки и формализации задач, умениям применять типовое прикладное программное обеспечение (в рамках начального курса ОИВТ).

Это позволит в дальнейшем не только полнее раскрыть научное, теоретическое содержание курса, но и эффективнее использовть (для развития всего школьного образования) широкий спектр связей ОИВТ с другими учебными предметами.

Что касается содержания второго этапа обучения информатике в средней школе, то пока это наиболее слаборазработанный компонент методической системы. На наш взгляд, отбор учебного материала, изучаемого на этом этапе, связан с его обобщением на основе фундаментальных понятий теории информатики, с более полным раскрытием сущности методов научного познания (моделирование, модельный и реальный эксперимент и т. д.), что должно обеспечить формирование у учащихся навыков использования компьютера как инструмента познавательной деятельности.

Важным вкладом в развитие современного научного мировоззрения школьников могло бы стать формирование при изучении курса ОИВТ в старших классах представлений об основных положениях кибернетики [4].

Совершенствование методов и организационных форм обучения

Говоря о совершенствовании методов и форм обучения информатике, в качестве одной из первоочередных мы должны назвать задачу преодоления разрыва между теоретическим и практическим аспектами содержания курса. По мере оснащения школ вычислительной техникой проблема организации практических занятий будет снята (по меньшей мере путем создания межшкольных кабинетов ВТ). Но одновременно это потребует создания новой методики проведения занятий, основанной на систематическом использовании ЭВМ на каждом уроке информатики.

К числу важнейших следует отнести и проблему выбора организационных **форм** проведения занятий. Анализ характера деятельности людей, занятых в информационной «индустрии», показывает, что ведущей здесь является **коллективная** форма деятельности. Учитывая необходимость передачи будущим выпускникам не только некоторой суммы знаний и умений, но и навыков работы в коллективе, следует шире применять такие формы работы учащихся, как семинары, учебные дискуссии, коллективно-распределенные формы работы с учебным материалом. Как показывает опыт, использование этих форм в преподавании других предметов (и предварительный эксперимент по применению их в преподавании информатики), методически обоснованное сочетание коллективных и индивидуальных форм организации учебной работы школьников при **ведущей** роли коллективной формы способны обеспечить лучшее усвоение программного материала, развитие самостоятельности и активности учащихся.

Специфика курса, связанная с широким применением компьютеров в процессе обучения, позволяет эффективно использовать в практике обучения систематическую текущую проверку знаний учащихся (на основе единых заданий, распространяемых на стандартном носителе — дискете), с автоматической обработкой результатов проверки и выдачей статистических результатов.

Совершенствование средств обучения

Одним из направлений совершенствования методики преподавания ОИВТ является создание **целостного учебно-методического комплекса** (УМК) по этому предмету. Сложность УМК по ОИВТ как системы, включающей в себя не только учебник и методическое пособие, но также систему книг для чтения, методические пособия по применению ППС, наконец, сами ППС, требует для своего создания привлечения широкого круга специалистов как в области педагогики, так и в области программирования.

Создание ППС (на современном этапе внедрения ВТ в обучение), особенно актуальное для преподавания ОИВТ, предполагает разработку концепции применения педагогических программных средств в учебном процессе. Эта концепция должна содержать не только анализ условий эффективного использования ППС для решения задач совершенствования обучения, но и **систему методических требований** к ППС, вытекающих из целей и содержания обучения ОИВТ. Особенно остро сегодня стоит проблема формулировки психолого-педагогических требований к ППС. Их разработка должна позволить избежать ошибок, характерных для внедрения ВТ в учебный процесс в школах Запада.

Перспективным, с нашей точки зрения, является подход к созданию учебного программного обеспечения, при котором учащимся раскрывается одна из важнейших функций компьютера — быть **инструментом** познавательной деятельности человека. Создание инструментальных программ для учащихся позволит сформировать у них навыки применения компьютеров в качестве средства повышения эффективности их учебной деятельности, что должно найти выражение в развитии их познавательной активности и самостоятельности.

Безусловно, используемые в обучении ППС должны обеспечивать не только процесс передачи знаний и формирования умений, но и оперативный контроль их усвоения учащимися, а также выдачу учителю интегрированной информации об уровне результатов обучения. Это предполагает создание системы диагностирующих программ, осуществляющих стандартную диагностику сформированности знаний, навыков и умений, предусмотренных программой курса.

Сравнение ОИВТ с другими учебными предметами показывает наличие в методической системе обучения этому предмету **специфического** компонента — компьютера как нового средства обучения.

Конечно, эта специфичность носит временный характер и с течением времени, по мере разработки методики использования учебного компьютера как средства обучения в других предметах, в основном будет утрачена. На сегодня же ОИВТ является единственным предметом, в рамках которого отрабатывается методика применения компьютеров в учебном процессе. Результаты, полученные в методике ОИВТ, со временем станут достоянием других учебных предметов. Специфичность же сохранится в том отношении, что в данном предмете компьютер будет выступать одновременно и как средство обучения, и как предмет изучения.

Перспективы переноса ОИВТ в V—VII классы

Перспективность переноса ОИВТ в младшие классы требует специального изучения как с точки зрения проверки возможности такого переноса и связанного с ним отбора учебного материала, подлежащего изучению, так и с точки зрения целесообразности, определяемой кругом решаемых при этом общеобразовательных (а не только узкопредметных) задач.

Лабораторией методики преподавания информатики и применения ЭВМ в учебном процессе НИИ СиМО АПН СССР разрабатывается концепция такого курса. В процессе ее разработки решается целый ряд задач:

выделение совокупности навыков и умений в области информатики, имеющих общеобразовательное значение и требующих формирования в более раннем возрасте (к их числу следует отнести и навыки применения компьютера);

определение содержания отдельных этапов формирования КГ, начиная с приобретения практических навыков обращения с ВТ и кончая использованием ее для решения учебных задач из разных предметов;

разработка методики формирования КГ, учитывающей возрастные возможности учащихся среднего школьного возраста;

исследование влияния работы с компьютером на интеллектуальное развитие школьников.

На протяжении трех лет в ряде школ проводится эксперимент по введению курса информатики в V—VII классах, в ходе которого, в частности, проверяется методическая целесообразность использования в обучении различных языковых средств (Бейсика, Пролога, ЛОГО) и специальных учебных программных средств (типа «система специализированных исполнителей»). Результаты этого и ряда других экспериментов, проводимых как в нашей стране, так и за рубежом, позволяют сделать заключение о возможности и целесообразности более раннего изучения основ информатики в средней школе.

Литература

1. *Ершов А. П.* Школьная информатика в СССР: от грамотности к культуре // Информатика и образование. 1987. № 6.

1. *Гершунский Б. С.* Компьютеризация в сфере образования: проблемы и перспективы. М.: Педагогика, 1987.

3. *Уваров А. Ю.* ЭВМ на пути в школу // Информатика и образование. 1986, № 1.

4. *Леднев В. С., Кузнецов А. А.* Перспективы изучения основ кибернетики в средней школе // Советская педагогика. 1975, № 6.

Солнечная музыка

С первыми лучами солнца в комнате одного из американских изобретателей начинает играть музыка и звучит до тех пор, пока не угаснет его последний луч. Прекрасное настроение на целый день создает автору изобретения его детище — музыкальный электронный автомат.

В маленькой плоской коробочке, на одну сторону которой наклеен солнечный элемент, находится микропроцессор и электронная память с записанной в нее заранее мелодией. Когда свет попадает на солнечный элемент, вырабатываемый им ток заставляет микропроцессор воспроизводить хранящуюся в запоминающем устройстве последовательность импульсов, которую пьезоэлемент, наклеенный на другую сторону коробочки, преобразует в колебания звуковой частоты.

Мелодию можно заменить, сменив элемент памяти, в котором она хранится, на новый.

Устройство герметично и не требует обслуживания. Встроенное в оконную раму или дачную дверь может служить бесконечно долго.

Е. ЛОДАТКО,
Славянский педагогический институт

Рекурсивные алгоритмы

Одним из наиболее эффективных приемов, используемых при разработке алгоритмов решения ряда задач, является применение вспомогательных алгоритмов. Вспомогательные алгоритмы — это алгоритмы, используемые в качестве составных частей в основном алгоритме. Особенности работы с такими алгоритмами рассмотрены в учебном пособии[1] и, в принципе, при достаточной детализации в процессе объяснения материала не вызывают существенных затруднений у учащихся.

С понятием вспомогательного алгоритма связано более сложное понятие *рекурсивного алгоритма*, т. е. алгоритма, использующего в качестве вспомогательного самого себя. Основой для разработки рекурсивных алгоритмов служат так называемые *рекуррентные соотношения* (формулы), устанавливающие зависимости между результатами каких-либо действий (операций) на n-м шаге от результатов аналогичных действий (операций), полученных на предыдущем $n-1$-м шаге.

Следует сказать, что знакомство учащихся даже с простейшими рекурсивными алгоритмами (вычисления факториала натурального числа и квадрата натурального числа), предусмотренное в курсе основ информатики и вычислительной техники, вызывает ряд трудностей в усвоении. Они обусловлены прежде всего сложностями структурно-логического характера в действиях, составляющих конструктивную основу таких алгоритмов. Естественно, подобные сложности и малый ресурс учебного времени не позволяют в процессе обучения специально уделять внимание вопросам разработки рекурсивных алгоритмов решения задач.

Вместе с тем эффективность таких алгоритмов в сочетании с их идейным богатством и логикой построения служат отправными моментами для включения в олимпиады по информатике задач, в основу решения которых положены рекурсии. Понятно, что для успешного решения подобных задач участники олимпиад должны кроме понимания сущности рекурсивных алгоритмов иметь также и определенный опыт их разработки, который не может дать разбор приведенных в учебном пособии задач. Чтобы учитель мог оказать помощь учащимся в более детальном знакомстве с рекурсивными алгоритмами как при организации консультаций для школьников, интересующихся предметом, так и при подготовке их к олимпиадам, рассмотрим некоторые задачи.

Для упрощения дальнейшего обсуждения решений задач обратимся вначале к известному примеру рекурсивного алгоритма вычисления факториала натурального числа, в основу которого положена рекуррентная формула:

$$k! = \begin{cases} 1, & \text{если } k=1 \\ (k-1)! \cdot k, & \text{если } k>1. \end{cases}$$

Сам алгоритм конструктивно прост.

```
алг нат ФАКТОРИАЛ (нат k)
нач
  если k=1
    то знач:=1
    иначе знач:=ФАКТОРИАЛ
    (k—1)·k
  все
кон
```

Логическая сложность его конструкции заключается в особенностях получения промежуточных результатов при последовательном обращении алгоритма к себе как вспомогательному. При этом важным для понимания самой идеи рекурсии является то, что в подобных алгоритмах можно выделить две серии шагов. Первая серия — это шаги рекурсивного «погружения» алгоритма в себя, т. е. по-

[1] Основы информатики и вычислительной техники: Пробное учебное пособие для средних учебных заведений. Ч. II / Под ред. А. П. Ершова, В. М. Монахова. М.: Просвещение, 1986.

следовательного обращения алгоритма к самому себе как вспомогательному до тех пор, пока выбранный параметр k не достигнет значения 1. Вторая серия — это шаги, обеспечивающие получение промежуточных и конечных результатов при последовательном выходе из рекурсий до начального значения k.

Такие серии шагов в простейших рекурсивных алгоритмах вычисления значений функций могут быть достаточно наглядно представлены с помощью таблиц[2], подобных приводимой ниже. В ней стрелками показано, как происходит рекурсивное «погружение» алгоритма в себя и как при последовательном выходе из рекурсий происходит формирование результата работы алгоритма при вычислении факториала числа 4.

Таблица 1

Значение k	4	3	2	1	Конечный результат и промежуточные результаты при выходе из рекурсий
Проверка условия ($k=1$?)	нет	нет	нет	да	
	ФАКТОРИАЛ (3)·4 →	ФАКТОРИАЛ (2)·3 →	ФАКТОРИАЛ (1)·2 →	1 →	знач:=1·2·3·4 знач:=1·2·3 знач:=1·2 знач:=1

Перейдем теперь к рассмотрению некоторых задач построения алгоритмов, эффективное решение которых основывается на применении рекурсий. При этом постараемся охватить не только алгоритмы вычисления значений функций, но и алгоритмы работы с графической и текстовой информацией.

Пример 1. Составить алгоритм нахождения натуральной степени вещественного числа.

Основываясь на рекуррентной формуле $x^k = x^{k-1} \cdot x$, можно разработать рекурсивный алгоритм, конструктивно подобный алгоритму вычисления факториала натурального числа.

```
алг вещ СТЕПЕНЬ (нат k, вещ x)
нач
    если k=1
        то знач:=x
        иначе знач:=СТЕПЕНЬ
        (k—1, x)·x
    все
кон
```

Работу этого алгоритма (на примере нахождения x^4) удобно, как и ранее, проиллюстрировать таблицей.

[2] См.: Исаков В. Исполнение алгоритмов // Информатика и образование. 1987. № 2. С. 30—43.

Таблица 2

Значение k	4	3	2	1	Конечный результат и промежуточные результаты при выходе из рекурсий
Проверка условия ($k=1$?)	нет	нет	нет	да	
	СТЕПЕНЬ (3, x)·x →	СТЕПЕНЬ (2, x)·x →	СТЕПЕНЬ (1, x)·x →	x →	знач:=$x\cdot x\cdot x\cdot x$ знач:=$x\cdot x\cdot x$ знач:=$x\cdot x$ знач:=x

Как видно из таблицы, работа этого алгоритма аналогична работе алгоритма вычисления факториала натурального числа, что обусловлено сходством рекуррентных формул. Естественно, приведенный пример допускает возможность построения другого алгоритма, в основу которого положен цикл **пока** или цикл с параметром **для...от...до**.

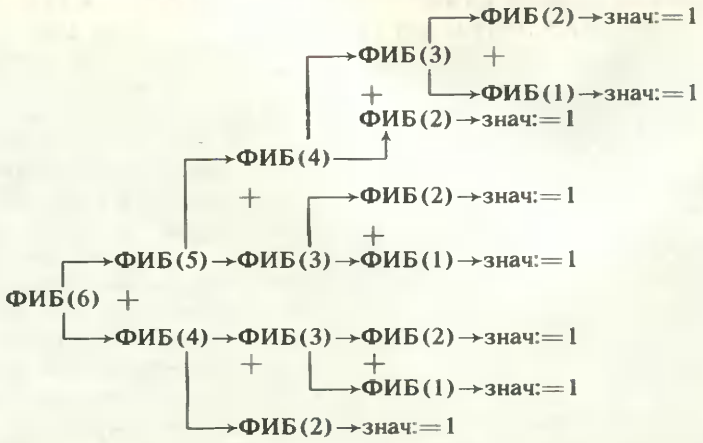
Пример 2. Числами Фибоначчи называются числа, удовлетворяющие соотношению $\Phi_k=\Phi_{k-1}+\Phi_{k-2}$, где $\Phi_1=1$, $\Phi_2=1$. Составить алгоритм нахождения числа Фибоначчи для любого натурального значения k.

Приведенное выше соотношение представляет собой рекуррентную формулу, позволяющую разработать рекурсивный алгоритм, поскольку каждое из чисел Φ_k, Φ_{k-1}, Φ_{k-2} также является числом Фибоначчи, а значит, определяется из того же соотношения, но лишь для меньших значений k.

```
алг нат ФИБОНАЧЧИ (нат k)
нач
    если k⩽2
        то знач:=1
        иначе знач:=ФИБОНАЧЧИ
        (k—1)+ФИБОНАЧЧИ    (k—2)
    все
кон
```

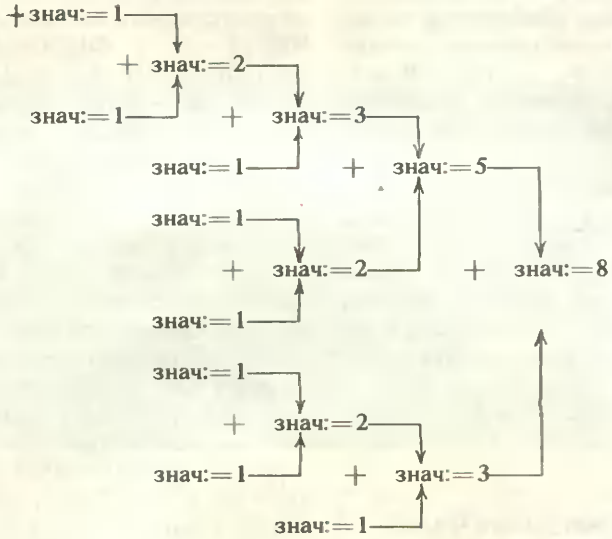
Работа этого алгоритма, в отличие от предыдущих, несколько сложнее, поскольку предполагает при невыполнении условия $k\leqslant 2$ «двойное» обращение алгоритма к самому себе как вспомогательному, обусловленное необходимостью нахождения Φ_{k-1} и нахождения Φ_{k-2}. В свою очередь, если значения $k-1$ и $k-2$ не удовлетворяют неравенствам $k-1\leqslant 2$ и $k-2\leqslant 2$ (т. е. условие $k\leqslant 2$ еще не выполняется), то каждый из «рекурсивных двойников» ФИБОНАЧЧИ $(k-1)$ и ФИБОНАЧЧИ $(k-2)$ приводит к необходимости обращения к самому себе как вспомогательному, т. е. к паре рекурсивных алгоритмов: ФИБОНАЧЧИ $(k-2)$ + ФИБОНАЧЧИ $(k-3)$ и ФИБОНАЧЧИ $(k-3)$ + ФИБОНАЧЧИ $(k-4)$. В результате этого шага работа алгоритма оказывается уже связанной с четырьмя алгоритмами ФИБОНАЧЧИ.

Если процесс будет продолжаться (в связи с тем что условие $k-i\leqslant 2$ еще не выполняется ни для одного из алгоритмов ФИБОНАЧЧИ $(k-i)$, полученных на очередном шаге рекурсии), то каждый из алгоритмов ФИБОНАЧЧИ $(k-i)$ в результате повторного обращения к себе как вспомогательному «порождает» пару алгоритмов, выполнение которых необходимо для выполнения алгоритма ФИБОНАЧЧИ на предыдущем шаге рекурсии. Общее число таких, требующих выполнения, алгоритмов на данном шаге рекурсии становится равным 8 и т. д. Но как только для какого-то из алгоритмов ФИБОНАЧЧИ $(k-i)$ перестанет выполняться условие $k-i\leqslant 2$, будет получен результат **знач**, равный единице, который в качестве промежуточного результата будет использован в дальнейшем при выходе из рекурсий и формировании конечного результата.



Граф рекурсивного «погружения» алгоритма в себя

Описанный процесс может быть достаточно наглядно представлен с помощью двух графов: графа рекурсивного «погружения» алгоритма в себя и графа получения результатов при выходе из рекурсий. Ниже они представлены для начального значения $k=6$ (для краткости название алгоритма дается в сокращенном виде: ФИБ).



Граф получения результатов при выходе из рекурсий

Следует отметить, что использованная при разработке алгоритма ФИБОНАЧЧИ рекурсивная конструкция дает возможность некоторых упрощений по сравнению с возможной циклической конструкцией этого алгоритма, в чем нетрудно убедиться из примера ниже.

```
алг ФИБОНАЧЧИ-ЦИКЛ (нат k, нат z)
арг k
рез z
нач нат x, y; цел i
    x:=1; z:=1
    если k>2
        то для i от 3 до k
            нц
                y:=x
                x:=z
                z:=x+y
            кц
    все
кон
```

Пример 3. Разработать рекурсивный алгоритм нахождения наибольшего числа среди K данных вещественных чисел.

В качестве основы для разработки такого алгоритма используем тот факт, что большее среди K чисел можно найти в результате сравнения K-го числа и большего среди K—1 чисел; в свою очередь, большее среди K—1 чисел можно найти в результате сравнения K—1-го числа и большего среди K—2 чисел и т. д. до сравнения 1-го и 2-го чисел. Используя этот факт, составим один из вариантов алгоритма.

```
алг вещ MAX (нат k, вещ таб A [1:k])
нач
    если k<2
        то знач:=A[k]
        иначе MAX (k—1, A[1:k—1])
            если знач ⩾ A[k]
                то знач:=A[k]
            все
    все
кон
```

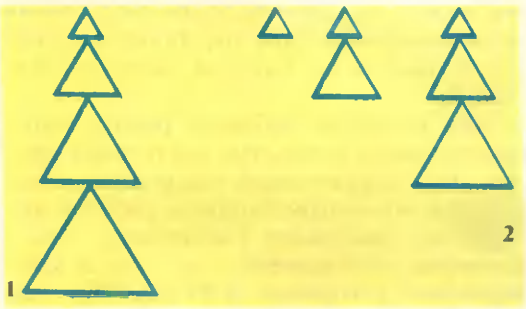
По конструкции алгоритм достаточно прост, однако, в отличие от алгоритмов, приведенных в ранее рассмотренных примерах, работает он сложнее, причем усложнения возникают при выходе алгоритма из рекурсий и связаны с необходимостью сравнения полученной на предыдущем шаге рекурсии величины знач с K-м числом и формирования нового промежуточного результата знач. Как происходит рекурсивное «погружение» алгоритма в себя и как происходит получе-

ние результатов, показано в таблице ниже[3].

В приведенной таблице в графе проверки условия знач⩽A[k] для наглядности вместо имени знач указаны числовые значения, которые оно принимает.

Обратимся теперь к примерам алгоритмов работы с графической информацией. Как известно, результатами работы таких алгоритмов являются рисунки, чертежи, схемы. Их построение при исполнении алгоритмов будем выполнять поэтапно, подобно тому как иллюстрировали получение результатов в алгоритмах вычисления значений функций с помощью таблиц.

Пример 4. Составить алгоритм рисования фигуры «елочка», изображенной на рисунке 1, для любого числа k ее звеньев (a — длина стороны наименьшего из звеньев, $k * a$ — длина стороны наибольшего из звеньев).



[3] Для конкретности таблицей иллюстрируется нахождение максимального элемента в таблице с элементами A[1]=3, A[2]=−2, A[3]=5,3, A[4]=4,8, A[5]=6.

Рассмотрим приемлемые варианты построения искомого алгоритма.

Учитывая, что «елочка» состоит из k одинаково рисуемых частей — правильных треугольников со сторонами a, $2a$, $3a$, ... ka, можно в конструкции алгоритма использовать цикл (вложенные циклы). Например, так, как показано ниже.

```
алг ЕЛОЧКА-ЦИКЛ (вещ a, нат k)
арг a, k
нач цел i, j
    для i от 1 до k
    нц
      рисуй
      для j от 1 до 3
      нц
        направо (120); вперед (i * a)
      кц
      не рисуй; налево (30); назад
      (i * a * 0,5 * √3)
    кц
кон
```

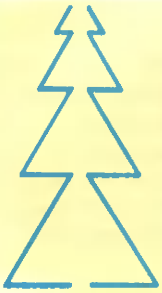

При исполнении этого алгоритма будут последовательно рисоваться части-треугольники, из которых состоит «елочка», причем с верхушки (рис. 2), т. е. начиная с самого маленького верхнего треугольника и до нижней k-й, наибольшей части.

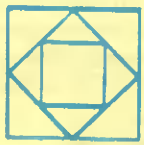
Менее очевидна возможная рекурсивная конструкция алгоритма рисования «елочки», при которой изображение формируется из двух «пилообразных» частей (рис. 3). Для рисования каждой из таких частей может применяться серия однотипных команд, позволяющая изображать любой «зуб» пилообразных частей, начиная с большего.

Таблица 3

Значения k	5	4	3	2	1	Проверка условия знач⩽A[k] (да/нет)	Конечный и промежуточный результаты при выходе из рекурсий
Проверка условия $k<2$	нет	нет	нет	нет	да		
	MAX(5,A[1:5])	MAX(4,A[1:4])	MAX(3,A[1:3])	MAX(2,A[1:2])	MAX(1,A[1:1])	A[3]⩽A[5]? (да) A[3]⩽A[4]? (нет) A[1]⩽A[3]? (да) A[1]⩽A[2]? (нет)	знач:=A[5] знач:=A[3] знач:=A[3] знач:=A[1] знач:=A[1]

3





4

Эта особенность рисования «елочки» и может рассматриваться в качестве рекуррентного соотношения — основы рекурсивного алгоритма.

```
алг ЕЛОЧКА (вещ a, нат k)
нач
    рисуй
    влево (90); вперед (0,5 * a * k)
    направо (120); вперед (a * k)
    налево (30)
    если k>1
        то ЕЛОЧКА (a, k—1)
    все
    направо (120); вперед (a * k)
    направо (120); вперед (0,5 * a * k)
    направо (90)
кон
```


Проиллюстрируем с помощью таблицы построение изображения «елочки» при исполнении приведенного алгоритма.

Как и ранее, выделим в таблице шаги рекурсивного погружения алгоритма в себя, а в графе результатов будем изображать те части рисунка, которые окажутся выполненными на очередном шаге алгоритма. Последовательность шагов и получения результатов будем указывать стрелками. Для определенности работы алгоритма положим $k=4$, т. е. покажем, как получается изображение «елочки», состоящее из четырех треугольников.

Пример 6. Составить алгоритм рисования фигуры «квадраты» (рис. 4)[4].

Так же как и в предыдущем примере, рисунок состоит из одинаково изображаемых частей (квадратов), положение которых относительно друг друга определяется поворотом на 45° и соотношением сторон $1:\frac{\sqrt{2}}{2}$.

Используя эту закономерность, можно разработать рекурсивный вариант алгоритма для построения K вписанных друг в друга квадратов со стороной a наибольшего из них.

```
алг КВАДРАТЫ (вещ a, нат k)
нач  цел i
    рисуй
    вперед (0,5 * a); направо (45)
    если k>1
        то КВАДРАТЫ (0,5 * √2 * a,
            k—1)
    все
    вперед (0,5  a)
    для i от 1 до 3
    нц
      направо (90); вперед (a)
    кц
    направо (45)
кон
```

Если обратиться к табличной иллюстрации работы приведенного алгоритма, то можно проследить этапы построения изображения квадратов, длина стороны наибольшего из которых принята равной 2.

Как видно из таблицы, работа этого рекурсивного алгоритма достаточно проста как в структурном, так и логическом плане и во многом сходна с работой алгоритма рисования «елочки». Такое сходство обусловлено тем, что в конструкции алгоритма КВАДРАТЫ, так же как и в алгоритме рисования «елочки», могут быть использованы вложенные циклы.

Обратимся теперь к примеру более сложного рекурсивного алгоритма, который дает действительное представление об эффективности подобных алгоритмов.

Пример 7. Фигура «снежинка» получается следующим образом: из одной точки-центра вырастают шесть кристалликов-отрезков длины a так, что любые два из них образуют угол, кратный 60°. К свободному концу каждого такого кристаллика-отрезка прикрепляется одним из свободных концов новая «снежинка» с длиной кристаллика-отрезка в K раз меньшей a, причем углы, образуемые любыми двумя кристалликами «снежинок», также кратны 60°. Аналогично к свободным концам кристалликов-отрезков вновь выросших «снежинок» прикрепляются новые «снежинки» с длиной

[4] Основы информатики и вычислительной техники: Пробное учебное пособие для средних учебных заведений. Ч. I / Под ред. А. П. Ершова, В. М. Монахова. М.: Просвещение, 1985. С. 81.

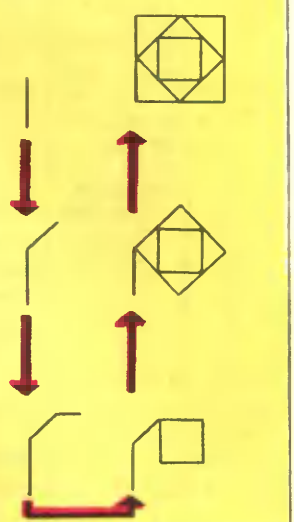
Таблица 4

Значения k	4	3	2	1	Последовательность шагов при выходе из рекурсий	Конечный и промежуточные результаты
Проверка условия $k=1$	нет	нет	нет	да		
	ЕЛОЧКА (1,4) рисуй влево (90) вперед (2) направо (120) вперед (4) налево (30)	ЕЛОЧКА (1,3) рисуй влево (90) вперед (1,5) направо (120) вперед (3) налево (30)	ЕЛОЧКА (1,2) рисуй влево (90) вперед (1) направо (120) вперед (2) налево (30)	ЕЛОЧКА (1,1) рисуй влево (90) вперед (0,5) направо (120) вперед (1) налево (30)	направо (120) вперед (4) направо (120) вперед (2) направо (90) направо (120) вперед (3) направо (120) вперед (1,5) направо (90) направо (120) вперед (2) направо (120) вперед (1) направо (90) направо (120) вперед (1) направо (120) вперед (0,5) направо (90)	

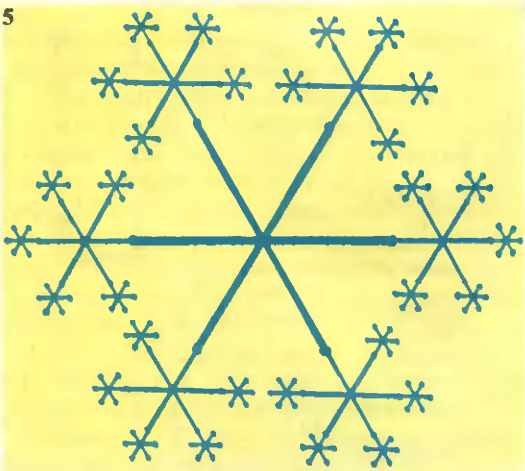
кристаллика в K раз меньше прежней. Указанный процесс продолжается M раз. Разработать алгоритм, рисующий «снежинку» для любых заданных значений a, K, M. (рис. 5)

Как видно из рисунка и следует из условия задачи, «снежинки», вырастающие на свободных концах другой «снежинки», могут изображаться при выполнении той же серии команд, с помощью которой была нарисована исходная «снежинка». В качестве такой серии команд целесообразно использовать цикл рисования шести кристалликов-отрезков «снежинки».

Вместе с тем если попытаться на ос-

Значения k	3	2	1	Последовательность шагов при выходе из рекурсий	Конечный и промежуточные результаты
Проверка условия $k=1$	нет	нет	да		
	КВАДРАТЫ (2,3) рисуй вперед (1) направо (45) →	КВАДРАТЫ (2,2) рисуй вперед (0,5*√2) направо (45) →	КВАДРАТЫ (1,1) рисуй вперед (0,5) направо (45) →	3 раза: направо (90) вперед (2); направо (45) ↑ 3 раза: направо (90) вперед (√2); направо (45) ↑ 3 раза: направо (90) вперед (1); направо (45) ↑	

5

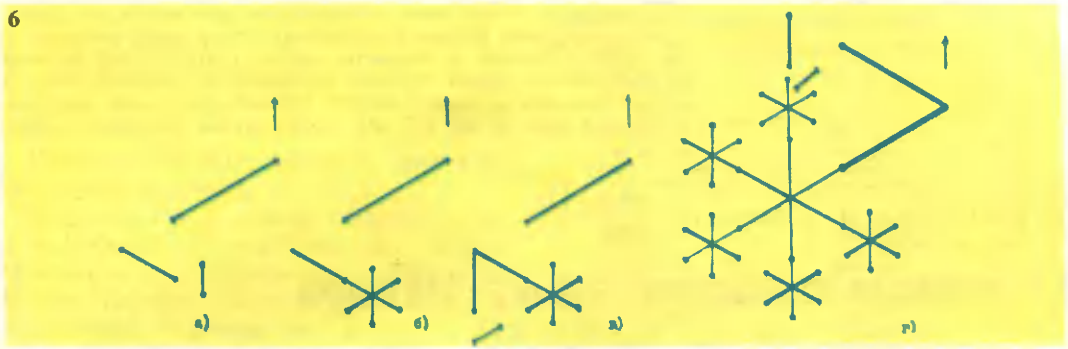


нове подобного цикла построить алгоритм, то окажется, что для различных значений М необходимо будет обращаться к вложению М раз циклов рисования «снежинки» друг в друга, т. е. фактически для каждого конкретного М нужно будет строить свой алгоритм, что явно не соответствует требованию задачи. Однако если в основу алгоритма положить рекурсивную конструкцию, позволяющую выполнять М вложений циклов рисования «снежинок», то указанный недостаток будет устранен. Сделать это можно так.

```
алг СНЕЖИНКА (вещ a, нат K, нат M)
нач цел i
    для i от 1 до 6
    нц
      рисуй
      налево (120); вперед (a)
      не рисуй
      вперед (a/K)
      если M>1
          то СНЕЖИНКА (a/K, K, M—1)
      все
      назад (a/K); налево (180)
      вперед (a)
    кц
кон
```

Особенностью работы приведенного алгоритма является то, что в нем в цикле осуществляется последовательный выбор кристалликов-отрезков, к свободным концам которых прикрепляются «последние» из вырастающих «снежинок». Как только такой конец оказывается найден, происходит построение изображения наименьшей «снежинки», после чего выбирается следующий кристаллик-

6



отрезок, к которому необходимо прикрепить «снежинку». Этапы поиска и рисования «снежинок» представлены для значений $a=4$, $K=2$, $M=3$ (рис. 6).

Заметим, что этот вариант рекурсивного алгоритма не является единственно возможным. Незначительное изменение описания процесса роста «снежинки» дает возможность разработать не менее интересный вариант алгоритма, использующего в качестве вспомогательного рекурсивный алгоритм.

```
алг СНЕЖИНКА-1 (вещ a, нат K, нат M)
арг a, K, M
нач цел i
    для i от 1 до 6
    нц
        ВЕТКА (a, K, M)
        вправо (120)
    кц
кон
алг ВЕТКА (вещ a, нат K, нат M)
нач цел j
    рисуй
        вперед (a)
        если M>1
            то для j от 1 до 5
                нц
                    вправо (120)
                    ВЕТКА (a/K, K, M—1)
                кц
                вправо (120)
            иначе вправо (180)
        все
        не рисуй
        вперед (a)
кон
```

Предоставляя читателю возможность самостоятельно выполнить рисунок «снежинки» с помощью этого алгоритма, заметим в заключение, что затронутая тема не является простой для понимания, однако при определенной методической обработке может стать основой для творчества учащихся с хорошей математической подготовкой в области конструирования различных алгоритмов.

КНИГИ

Издательство «Наука» в 1988 г. выпустило сборник статей «Информатика и компьютерная грамотность» под редакцией академика Б. Н. Наумова.

В сборнике рассматриваются вопросы, связанные с содержанием учебной информатики как в теоретическом, так и в практическом аспекте. Обсуждаются концепции компьютеризации образования, зарубежный опыт, перспективы развития учебной информатики в нашей стране и за рубежом. Ряд статей посвящен программно-аппаратным средствам и программному обеспечению учебного назначения, проблемам унификации и стандартизации в области учебной информатики, применению различных языков программирования для учебных целей.

Обсуждается конкретный опыт преподавания информатики в школе, учебно-производственных центрах, средних специальных учебных заведениях, вузах, при подготовке и переподготовке административных кадров и т. д.

Сборник будет полезен как специалистам в области учебной информатики, так и тем, кто впервые сталкивается с проблемами использования компьютеров в сфере образования.

Книгу можно заказать в магазинах «Академкнига». Адрес магазина «Книга — почтой» центральной конторы «Академкнига»: 117192, Москва, Мичуринский проспект, 12. Тираж 11 000 экз.

Мы начинаем публикацию методических материалов по языку программирования Рапира. Описываемая версия языка реализована на ПЭВМ «Ямаха» и «Корвет», однако статьи будут полезны и обладателям других ПЭВМ. Замечания и вопросы авторам можно посылать по адресу: 630090, Новосибирск, просп. Академика Лаврентьева, 6, ВЦ СО АН СССР, группа школьной информатики.

Н. ГЛАГОЛЕВА, Н. ЮНЕРМАН,
г. Новосибирск

Основные понятия языка Рапира

Язык программирования, как и любой естественный язык, состоит из предложений. В Рапире эти предложения называются *предписаниями*. Они задают действия над данными. Простейшие примеры данных — числа и тексты.

Числа и операции над ними

Запись целых чисел в Рапире не отличается от общепринятой, а при записи вещественных чисел целая часть от дробной отделяется точкой, а не запятой, как принято в математике. Пример: 0.83, 3.1415. Над числами можно выполнять следующие операции:

\+ сложение,
— вычитание,
* умножение,
/ деление,
// деление нацело (25//4=6; (—25)//4=—7)*,
/% остаток от деления нацело (25/%4=1; (—25)/%4=3),
** возведение в степень.

Сравните математическую форму записи числовых выражений с их записью на Рапире.

$99+3\times5+12:4$	99+3*5+12/4
$-(4\times5):8$	—(4*5)/8
$\frac{6+12}{3+9}$	(6+12)/(3+9)
$2^3-8\times2$	2**3—8*2
$\frac{15}{2\times3}$	15/(2*3)

Обратите внимание, что любое выражение на Рапире записывается в строчку. Порядок выполнения действий в выражении такой. Сначала выполняются операции возведения в степень, после этого — операции умножения, деления, деления нацело и нахождения остатка от деления (в порядке их записи слева направо), в последнюю очередь выполняются операции сложения и вычитания. Для изменения порядка действия применяются только круглые скобки. Знак умножения опускать нельзя. Не допускается также использование других обозначений для знаков операций, кроме описанных.

Задача 1.

а) Найдите значения выражений:

(—3)**2—4*2*(—5);
—(12/2)+(1+2)**3—11;
8*64//7;
49//8+57/%4**2.

б) Запишите выражения по правилам языка Рапира:

$8(1+20:10)^2$

$\frac{(-8)^3+15^3}{-8+15}$

$(7^2+3^2-2\times7\times3)\,(7^2+3^2+2\times7\times3)$

$12\times7^3-8$

Тексты и операции над ними

Текстом называется любая последовательность символов. Чтобы отличать текст в программах, его заключают в кавычки (эти кавычки в текст не входят). Примеры текстов:

«черепаха»
«10-В КЛАСС»
«1988»
«%5. Хи-хи»
«Я на солнышке лежу»
«» (пустой текст)

Количество символов в тексте называется *длиной текста*. Операция определения длины текста обозначается знаком # .

* О делении нацело отрицательных чисел см.: Егоров А. А. Сравнение по модулю и арифметика остатков // Квант. 1970. № 5.

Например:

«КРОКОДИЛ» =8
«МИРУ — МИР»=10
«␣» =1*
« » =0

Результат этой операции — натуральное число или ноль.

Все символы в тексте считаются занумерованными слева направо. Номер символа в тексте называется *индексом*. Индекс первого символа равен единице, последнего — длине текста.

Выборка — операция извлечения (выбора) из текста одного символа по его индексу. Индекс выбираемого символа записывается в квадратных скобках после текста. Результат этой операции — текст из одного символа.

Например:

«КВАНТ»[3] =«А»
«КВАНТ»[#«КВАНТ»]=«Т»

Вырезка — операция выделения («вырезания») участка текста. Для выполнения этой операции нужно указать два индекса — номера начального и конечного символов вырезаемого участка текста. Эти номера записываются после текста в квадратных скобках и разделяются двоеточием. Результат этой операции — текст. Например: «черепаха»[3:6]=«репа».

При записи операции вырезки можно не указывать начальный индекс (тогда он считается равным 1) или конечный (тогда конечный индекс считается равным длине текста). Так, записи «ПОБЕДА»[3:6] и «ПОБЕДА»[3:] равносильны.

Тексты можно склеивать. Операция *склеивания* текстов обозначается знаком +. При ее выполнении к первому тексту дописывается второй. Результат — снова текст. Например, «лимон»+«ад»=«лимонад».

Текст можно *умножать* на натуральное число или ноль. Например, 2*«КВА»=«КВАКВА». Этот же результат можно получить, выполнив «КВА»*2. Результатом умножения текста на ноль является пустой текст. Например: «информатика»*0=«».

Рассмотрим примеры текстовых выражений, порядок выполнения операций в них и определим значения этих выражений.

```
   1) "МЕД" + "ВЕДАЮ"[:3] + "Ь"
"ВЕДАЮ"[:3]                    "ВЕД"
"МЕД" + "ВЕД" + "Ь"            "МЕДВЕДЬ"
"МЕД" + "ВЕДАЮ"[:3] + "Ь" = "МЕДВЕДЬ"
   2) "ДРОВА"[:#"ДРОВА"-1] + "О" + "СЕЧЬ"[:2] + "К"
#"ДРОВА"                       5
5-1                            4
"ДРОВА"[:4]                    "ДРОВ"
"СЕЧЬ"[:2]                     "СЕ"
"ДРОВ" + "О" + "СЕ" + "К"      "ДРОВОСЕК"
"ДРОВА"[:#"ДРОВА"-1]+"О"+"СЕЧЬ"[:2]+"К"="ДРОВОСЕК"
   3) 2 * "ПРАДЕДУШКА"[:3] + "БАБУШКА"
"ПРАДЕДУШКА"[:3]               "ПРА"
2 * "ПРА"                      "ПРАПРА"
"ПРАПРА" + "БАБУШКА"           "ПРАПРАБАБУШКА"
2 * "ПРАДЕДУШКА"[:3] + "БАБУШКА" = "ПРАПРАБАБУШКА"
   4) (2*"МЫШКА")[4:8]
2*"МЫШКА"                      "МЫШКАМЫШКА"
"МЫШКАМЫШКА"[4:8]              "КАМЫ"
(2*"МЫШКА")[4:8] = "КАМЫ"
```

Значением текстового выражения является сам текст (без кавычек). Например, текст 5*5= является значением выражения «5*5=».

Задача 2.

а) Найдите значения выражений:
«КАРАУЛ»[3]+«КАРАУЛ»[5]+«КАРАУЛ»[:2]
«ИНФОРМАЦИЯ»[:5]+«МАТЕМАТИКА»[5:]
«ПАРОВОЗ»[:4]+«ХОД»
2*«КУ»+«ШКАФ»[:3]

б) Дан текст ЧЕРЕПАХА. Используя только этот текст и операции над текстами, запишите выражения, значениями которых являются:
тексты ЧЕХ, ПАРЧА, ПАПАХА;
число 8;
число 3.

Предписание вывода

Значение выражения можно напечатать (выдать) с помощью предписания вывода. Для этого надо написать ВЫВОД: и после двоеточия указать через запятую все выражения, значения которых надо выдать.

Рассмотрим примеры предписаний вывода и что будет напечатано после выполнения каждого из них.

1) ВЫВОД:5
   5
2) ВЫВОД:2,6*7
   2 42
3) ВЫВОД:«5*5=», 5*5
   5*5=25
4) ВЫВОД:«МАШИНА»,«АГАТ»
   МАШИНААГАТ

* Знак ␣ показывает, что здесь стоит символ «пробел».

5) ВЫВОД:
«ПРОГРАММИРОВАНИЕ»[:8]+«А»
ПРОГРАММА

Каждое предписание вывода дает ЭВМ указание печатать с новой строки. Значения выражений, записанных в одном предписании вывода, выдаются на одной строке друг за другом, а перед числами ставится пробел.

Задача 3. Измените предписание 4 так, чтобы слова МАШИНА и АГАТ были напечатаны на одной строке отдельно друг от друга.

Блоки памяти и имена

Будем представлять память вычислительной машины в виде отдельных блоков. В каждом таком блоке может храниться один объект (число, текст, программа или ее часть и т. п.). В одном и том же блоке сначала может храниться, например, число, а потом — текст или программа. Блоку памяти можно дать *имя*, которое позволяет обратиться к хранящемуся в блоке объекту. Содержимое блока памяти называется *значением имени*. Выделение блоков памяти для хранения информации выполняется машиной. Имена для блоков придумывает программист и записывает их по правилам языка программирования. *Имя* в Рапире — это любая последовательность букв, цифр и знака _ (подчеркивание), начинающаяся с буквы. Примеры имен: J, К174, ТУ_134_Б, ПЛАНЕТА_ЗЕМЛЯ, SU112.

Последовательность символов _КЛАСС_10 не является именем, так как имя должно начинаться с буквы.

Задача 4. Являются ли именами: 10_В; А*4; ПИ; СЧЕТЧИК; А9.8? Объясните почему.

Предписание присваивания

Выделить блок памяти и поместить в него значение можно с помощью предписания присваивания. Для этого в левой части предписания присваивания записывается имя, которому будет присвоено значение, затем знак := (читается «присвоить») и в правой части — выражение, значение которого должно быть присвоено указанному имени.

Вот примеры предписаний присваивания:

```
ДЛИНА:=8
ШИРИНА:=7
Т:=«МАЛЫШ И КАРЛСОН»
```

Как выполняется предписание присваивания? Сначала вычисляется значение выражения и проверяется, есть ли в памяти блок с указанным именем. Если такого блока нет, то выделяется новый блок, ему дается это имя и в блок помещается вычисленное значение. Если в памяти есть блок с указанным именем, то старое содержимое этого блока уничтожается и на его место помещается новое значение.

Имена могут использоваться в выражениях. Например:

```
(ДЛИНА+ШИРИНА)*2.
```

В таких случаях при нахождении значения выражения вместо всех имен подставляются их значения.

Запишем в виде таблицы имена и те значения, которые они будут иметь после выполнения следующих предписаний.

```
ДЛИНА:=8
ШИРИНА:=7
ПЕРИМЕТР:=(ДЛИНА+ШИРИНА)*2
ПЛОЩАДЬ:=ДЛИНА*ШИРИНА
Т:=«МАЛЫШ И КАРЛСОН»
```

Имя	Значение имени
ДЛИНА	8
ШИРИНА	7
ПЕРИМЕТР	30
ПЛОЩАДЬ	56
Т	МАЛЫШ И КАРЛСОН

Такая таблица называется таблицей значений имен.

Рассмотрим программу (последовательность предписаний):

```
ДЛИНА:=8
ШИРИНА:=7
ВЫВОД:«ШИРИНА=»,ШИРИНА,« М»
ВЫВОД:«ДЛИНА=»,ДЛИНА,« М»
ВЫВОД:«ПЕРИМЕТР=»,(ШИРИНА+
                    ДЛИНА)*2,« М»
ВЫВОД:«ПЛОЩАДЬ=»,ДЛИНА*
                    ШИРИНА,« КВ.М»
```

В результате выполнения этой программы будет выдано:

```
ШИРИНА= 7 М
```

```
ДЛИНА= 8 М
ПЕРИМЕТР= 30 М
ПЛОЩАДЬ= 56 КВ.М
```

Задача 5. Дана программа:

```
ВРЕМЯ:=35
СКОРОСТЬ:=66
ПУТЬ:=(ВРЕМЯ*СКОРОСТЬ)/60
ВЫВОД:«ВРЕМЯ=»,ВРЕМЯ,« МИН»
ВЫВОД:«СКОРОСТЬ=»,СКОРОСТЬ,
                        «КМ/Ч»
ВЫВОД:«ПУТЬ=»,ПУТЬ,« КМ»
```

Напишите, что будет выдано после ее выполнения.

Рассмотрим программу:

```
ВОЗРАСТ:=13
ВЫВОД:«ВАМ СЕЙЧАС»,ВОЗРАСТ,« ЛЕТ»
ВОЗРАСТ:=ВОЗРАСТ+3
ВЫВОД:«ЧЕРЕЗ 3 ГОДА ВАМ БУДЕТ»,
      ВОЗРАСТ,« ЛЕТ»
```

При выполнении первого предписания присваивания выделяется (или разыскивается) блок памяти с именем ВОЗРАСТ и в него помещается число 13. Затем выдается сообщение

ВАМ СЕЙЧАС 13 ЛЕТ

При выполнении второго предписания присваивания сначала разыскивается блок с именем ВОЗРАСТ, затем к его содержимому прибавляется число 3, полученное значение присваивается имени ВОЗРАСТ (помещается в блок памяти с этим именем). Исполнение программы завершается выводом сообщения

ЧЕРЕЗ 3 ГОДА ВАМ БУДЕТ 16 ЛЕТ

Таблица значений имен в процессе исполнения программы изменяется так:

Имя	Значение имени
ВОЗРАСТ	~~13~~ 16

Слева от знака присваивания после имени могут быть записаны операции выборки и вырезки. Пусть, к примеру, в блоке с именем ГОРОД хранится текст ЕКАТЕРИНОДАР. Давайте разберемся, что произойдет с этим текстом после выполнения предписания

```
ГОРОД[:7]:=«КРАС»
```

Из содержимого блока ГОРОД (т. е. текста ЕКАТЕРИНОДАР) будет вырезан кусочек ЕКАТЕРИ и заменен на текст КРАС. После этого значением имени ГОРОД станет текст КРАСНОДАР.

Соотношение между длинами заменяемых текстов может быть произвольным. В нашем примере длинный текст (ЕКАТЕРИ) заменяется на короткий (КРАС).

Когда надо заменить один символ ровно на один, лучше воспользоваться операцией выборка. Например:

```
КУСТ:=«КАЛИНА»
КУСТ[1]:=«М»
```

После выполнения этих предписаний значением имени КУСТ станет текст МАЛИНА.

Задача 6. Придумайте аналогичные примеры, когда:

а) короткий текст заменяется на длинный;

б) заменяются тексты равной длины.

Рассмотрим программу:

```
СЛОВО:="ЧЕРЕПАШКА"; ВЫВОД:"СНАЧАЛА БЫЛА ", СЛОВО
СЛОВО[3:5]:="БУР" ; ВЫВОД:"А ТЕПЕРЬ ЕСТЬ ", СЛОВО
СЧИТАЛКА:="...КОРОЛЬ,КОРОЛЕВИЧ,САПОЖНИК,ПОРТНОЙ ..."
ТИТУЛ:=СЧИТАЛКА[12:18] + СЧИТАЛКА[24]; ВЫВОД:ТИТУЛ
ТИТУЛ[5:6]:=""; ВЫВОД:ТИТУЛ
ТИТУЛ[5]:="Н"; ВЫВОД:ТИТУЛ
```


Несколько предписаний могут располагаться на одной строке. В этом случае они разделяются точкой с запятой.

Выполним *ручную прокрутку* этой программы — составим таблицу значений имен и запишем, что будет выдано после выполнения программы.

ТАБЛИЦА ЗНАЧЕНИЙ ИМЕН

ИМЯ	ЗНАЧЕНИЕ ИМЕНИ		
СЛОВО	~~ЧЕРЕПАШКА~~	ЧЕБУРАШКА	
СЧИТАЛКА	...КОРОЛЬ, КОРОЛЕВИЧ, САПОЖНИК, ПОРТНОЙ ...		
ТИТУЛ	~~КОРОЛЕВА~~	~~КОРОВА~~	КОРОНА

Будет выдано:

```
СНАЧАЛА БЫЛА ЧЕРЕПАШКА
А ТЕПЕРЬ ЕСТЬ ЧЕБУРАШКА
КОРОЛЕВА
КОРОВА
КОРОНА
```

Рассмотрим следующую задачу. Пусть время суток задается парой чисел (часы, минуты). Мальчик собирается в воскресенье сходить на речку и в кино. Из дому он хотел бы выйти в половине десятого утра, а чтобы успеть в кино, он должен вернуться домой в 14 ч 10 мин. На дорогу от дома до речки и обратно он обычно тратит 1 ч 10 мин. Сколько времени (в часах и минутах) мальчик может провести на реке?

Сначала выберем имена для размещения начальных данных. При этом будем стремиться отразить в имени характер содержимого (так называемое правило мнемоничности).

ЧАС_ВЫХ — час выхода на речку
МИН_ВЫХ — минута выхода на речку
ЧАС_КИНО — час возвращения домой перед кино
МИН_КИНО — минута возвращения домой перед кино
ЧАС_ДОР — часы на дорогу до реки и обратно
МИН_ДОР — минуты на дорогу до реки и обратно

Выразим исходные данные в минутах:
время выхода из дому равно
ЧАС_ВЫХ*60+МИН_ВЫХ;
время возвращения домой равно
ЧАС_КИНО*60+МИН КИНО;
время на дорогу равно
ЧАС_ДОР*60+МИН_ДОР.

Значения этих выражений присвоим соответственно именам
ВРЕМЯ_ВЫХ, ВРЕМЯ_КИНО и ВРЕМЯ_ДОР.

Составим теперь выражение для нахождения искомого времени.
ВРЕМЯ_КИНО — ВРЕМЯ_ВЫХ — ВРЕМЯ_ДОР

Составим программу для определения времени пребывания на реке.

```
ЧАС_ВЫХ:=9;   МИН_ВЫХ:=30              \ НАЧАЛЬНЫЕ ДАННЫЕ
ЧАС_КИНО:=14; МИН_КИНО:=10             \ НАЧАЛЬНЫЕ ДАННЫЕ
ЧАС_ДОР:=1;   МИН_ДОР:=10              \ НАЧАЛЬНЫЕ ДАННЫЕ
ВРЕМЯ_ВЫХ:=ЧАС_ВЫХ * 60 + МИН_ВЫХ     \ ВЫХОД НА РЕЧКУ
ВРЕМЯ_КИНО:=ЧАС_КИНО * 60 + МИН_КИНО   \ ВЫХОД В КИНО
ВРЕМЯ_ДОР:=ЧАС_ДОР * 60 + МИН_ДОР      \ ВРЕМЯ НА ДОРОГУ
НА_РЕКЕ:=                              \ ВРЕМЯ НА РЕКЕ
  ВРЕМЯ_КИНО - ВРЕМЯ_ВЫХ - ВРЕМЯ_ДОР   \ В МИНУТАХ
ЧАС_РЕКА:=НА_РЕКЕ//60                  \ ЧАСЫ НА РЕКЕ
МИН_РЕКА:=НА_РЕКЕ/%60                  \ МИНУТЫ НА РЕКЕ
ВЫВОД:"НА РЕКЕ МОЖНО ПРОВЕСТИ:"        \ ВЫВОД РЕЗУЛЬТАТА
ВЫВОД:ЧАС_РЕКА," ЧАС."                 \ ВЫВОД РЕЗУЛЬТАТА
ВЫВОД:МИН_РЕКА," МИН."                 \ ВЫВОД РЕЗУЛЬТАТА
```

В конце каждой строки после символа записаны комментарии — пояснения к тексту программы. Они помогают лучшему пониманию текста программы человеком и не влияют на ее выполнение машиной.

Задача 7. Выполните ручную прокрутку этой программы.

Решения задач

Задача 1.

а) (—3)**2—4*2*(—5)=49
—(12/2)+(1+2)**3—11=10
8*64//7=73
49//8+57/%4**2=15

б) 8*(1+20/10)**2
((—8)**3+15**3)/(—8+15)
(7**2+3**2—2*7*3)*(7**2+3**2+2*7*3)
12*7**3—8

Наиболее распространенные ошибки и недочеты:

нарушен порядок действий;

неверно найдены результаты отдельных операций (особенно //, /% и **);

запись выражения не в одну строчку;

пропуск знака * при записи операции умножения;

пропуск скобок или их неправильная расстановка.

Задача 2.

а) «КАРАУЛ»[3]+«КАРАУЛ»[5]+«КАРАУЛ»[:2]=«РУКА»
«ИНФОРМАЦИЯ»[:5]+«МАТЕМАТИКА»[5:]=«ИНФОРМАТИКА»
«ПАРОВОЗ»[:4]+«ХОД»=«ПАРОХОД»
2*«КУ»+«ШКАФ»[:3]=«КУКУШКА»

Допускается ответ в виде списка значений. Тогда он должен выглядеть так:
РУКА
ИНФОРМАТИКА
ПАРОХОД
КУКУШКА

Наиболее распространенные ошибки и недочеты:

пропущены кавычки, когда ответ записан в виде равенств текстовых выражений;

поставлены кавычки, когда ответ дан в виде списка значений;

неправильно определены значения отдельных слагаемых.

б) Текст ЧЕХ. Возможные решения:
1) «ЧЕРЕПАХА»[:2]+«ЧЕРЕПАХА»[7]
2) «ЧЕРЕПАХА»[1:2]+«ЧЕРЕПАХА»[7]
3) «ЧЕРЕПАХА»[1]+«ЧЕРЕПАХА»[2]+«ЧЕРЕПАХА»[7]
4) «ЧЕРЕПАХА»[1]+«ЧЕРЕПАХА»[4]+«ЧЕРЕПАХА»[7]

Равносильными являются выражения: первое и второе, третье и четвертое. При этом третье и четвертое решения являются нерациональными.

Текст ПАРЧА. Возможные решения:
1) «ЧЕРЕПАХА»[5:6]+«ЧЕРЕПАХА»[3]+«ЧЕРЕПАХА»[1]+«ЧЕРЕПАХА»[6]

2) «ЧЕРЕПАХА«[5:6]+«ЧЕРЕПАХА»[3]+
«ЧЕРЕПАХА»[1]+«ЧЕРЕПАХА»[8]

Эти решения равносильны. Правильными, но не рациональными являются, например, следующие:

1) «ЧЕРЕПАХА»[5]+«ЧЕРЕПАХА»[6]+
«ЧЕРЕПАХА»[3]+«ЧЕРЕПАХА»[1]+
«ЧЕРЕПАХА»[6]
2) «ЧЕРЕПАХА»[5]+«ЧЕРЕПАХА»[8]+
«ЧЕРЕПАХА»[3]+«ЧЕРЕПАХА»[1]+
«ЧЕРЕПАХА»[6]

Текст ПАПАХА. Возможные решения:

2*«ЧЕРЕПАХА»[5:6]+«ЧЕРЕПАХА»[7:8]
«ЧЕРЕПАХА»[5:6]*2+«ЧЕРЕПАХА»[7:8]

Эти два решения равносильны. Правильным, но менее рациональным является решение

«ЧЕРЕПАХА»[5:6]+«ЧЕРЕПАХА»[5:6]+
«ЧЕРЕПАХА»[7:8]

Число 8 является значением выражения #«ЧЕРЕПАХА».

Число 3 является, например, значением выражения #«ЧЕРЕПАХА»[:3] (возможна вырезка любого трехсимвольного фрагмента текста).

Наиболее распространенные ошибки и недочеты:

текст пишут только в одном слагаемом, а затем указывают только индексы, например: «ЧЕРЕПАХА»[5:6]+[7:8];

переносят знак + при переходе на новую строчку или ставят знак переноса в тексте ЧЕРЕПАХА;

для соседних символов вместо одной операции вырезки используют несколько операций выборки.

Задача 3. Равносильными являются следующие решения:

ВЫВОД: «МАШИНА АГАТ»
ВЫВОД: «МАШИНА», «_», «АГАТ»
ВЫВОД: «МАШИНА_», «АГАТ»
ВЫВОД: «МАШИНА», «_АГАТ»

Задача 4.

10_В — нет (начинается с цифры);
А*4 — нет (символ * — не буква, не цифра и не знак _);
ПИ — да;
СЧЕТЧИК — да;
А9.8 — нет (символ .— не буква, не цифра и не знак _).

Задача 5. После исполнения программы будет выдано:

ВРЕМЯ= 35 МИН
СКОРОСТЬ= 66 КМ/Ч
ПУТЬ= 38.5 КМ

Обратите внимание на:

наличие знака «_» перед числами и перед вторыми текстами;

распределение по строчкам;

на всех строках вывод начинается с одной позиции.

Задача 6. Наиболее интересные решения получаются при использовании осмысленных текстов. Приведем по одному из возможных решений для каждого случая.

а) Замена короткого текста на длинный. Текст ТЕЛЕФИЛЬМ преобразуется в текст ТЕЛЕСПЕКТАКЛЬ.

ЖАНР:=«ТЕЛЕФИЛЬМ»
ЖАНР[5:]:=«СПЕКТАКЛЬ»

б) Замена текстов равной длины. Текст ТЕЛЕВИЗОР преобразуем в текст ТЕЛЕЦЕНТР.

СЛОВО:=«ТЕЛЕВИЗОР»
СЛОВО[5:]:=«ЦЕНТР»

Правильным является решение, в котором вместо первого из предписаний присваивания записано, что исходный текст хранится в блоке с некоторым именем.

Наиболее распространенные ошибки и недочеты:

пропущены кавычки в записи текстов в предписаниях присваивания;

слева от знака присваивания вместо имени с вырезкой стоит текст, за которым следует операция вырезки.

Задача 7.

Таблица значений имен имеет вид:

Имя	Значение имени
ЧАС_ВЫХ	9
МИН_ВЫХ	30
ЧАС_КИНО	14
МИН_КИНО	10
ЧАС_ДОР	1
МИН_ДОР	10
ВРЕМЯ_ВЫХ	570
ВРЕМЯ_КИНО	850
ВРЕМЯ_ДОР	70
НА_РЕКЕ	210
ЧАС_РЕКА	3
МИН_РЕКА	30

Будет выдано:
НА РЕКЕ МОЖНО ПРОВЕСТИ
3 ЧАС.
30 МИН.

Продолжение следует

Н. ПЕРЕВЕРЗЕВА, И. РЕВЧУК

Арифметические задачи на алгоритмическом языке

Широкие возможности для построения интересных и содержательных алгоритмов открывают задачи, связанные с нахождением натуральных чисел по известным признакам, с решением уравнений в целых числах, с проблемами делимости и т. д.

Приведем несколько примеров подобных задач с решениями, записанными на алгоритмическом языке.

Задача 1. Сумма цифр двухзначного числа равна 11. Если к этому числу прибавить 27, то получится число, записанное теми же цифрами, но в обратном порядке. Составить алгоритм поиска этого числа, если оно существует.


```
алг пример1 (цел X, лит T)
 дано
 надо X, T
нач цел A, Б, АБ, БА
. T := 'нет числа'
. A := 2
. нц пока T = 'нет числа' и A <= 9
. . Б := 11 - A
. . АБ := 10 * A + Б
. . БА := 10 * Б + A
. . если АБ + 27 = БА
. . .то X := АБ
. . .  T := 'есть число'
. . все
. . A := A + 1
. кц
кон
```

Задача 2. Сумма квадратов цифр некоторого двузначного числа на 1 больше утроенного произведения этих цифр. После деления этого двузначного числа на сумму его цифр в частном получится 7 и в остатке 6. Составить алгоритм поиска этого числа, если оно существует.

```
алг пример2 (цел X, лит T)
 дано
 надо X, T
нач цел A, Б, C
. T := 'пусто';  A := 0
. нц пока T <> 'есть решение' и A <= 9
. . A := A + 1 ; Б := 0
. . нц пока T <> 'есть решение' и Б <= 9
. . . Б := Б + 1
. . . C := 10 * A + Б
. . . если A*A+Б*Б=3*A*Б+1 и C=(A+Б)*7+6
. . . .то X := C
. . . . T := 'есть решение'
. . . все
. . кц
. кц
кон
```

Задача 3. Составить алгоритм, определяющий количество пар целых положительных чисел X, Y, удовлетворяющих уравнению $AX+BY=C$, где A, B, C — натуральные числа, $C>max(A, B)$.

```
алг пример3 (вещ A, B, C, цел K)
 дано A, B, C = 7, 125, 767
 надо K
нач цел X, Y, вещ ГРХ, ГРУ
. K := 0
. ГРХ := (C - B)/A
. ГРУ := (C - A)/B
. нц для X от 1 до ГРХ
. . нц для Y от 1 до ГРУ
. . . если A * X + B * Y = C
. . . .то K := K + 1
. . . все
. . кц
. кц
кон
```

Задача 4. Составить алгоритм, вычисляющий количество трехзначных натуральных чисел, сумма цифр которых равна их произведению.

```
алг пример4 (цел K)
 дано
 надо K
```

```
нач цел A, B, C
. K := 0
. нц для A от 1 до 9
. . нц для B от 1 до 9
. . . нц для C от 1 до 9
. . . . если A ≠ B ≠ C=A + B + C
. . . . .то K := K + 1
. . . . все
. . . кц
. . кц
. кц
кон ----------------------------------------
```

Задача 5. Даны два натуральных числа A и B. Составить алгоритм, определяющий, является ли одно из них делителем другого.

```
алг пример5 (цел A, B, лит C)
 дано A, B = 34, 68
 надо C
нач
. если ОСТ(A,B)=0 или ОСТ(B,A)=0
. .то C := 'да'
. .иначе C := 'нет'
. все
кон ----------------------------------------
```

Задача 6. Даны два натуральных числа M и K. Составить алгоритм, определяющий, на какую максимальную степень числа K делится M.

```
алг пример6 (цел M, K, C)
 дано M, K = 512, 2
 надо C
нач   лит признак
. признак := 'считай'
. C := 0
. нц пока признак = 'считай'
. . если ОСТ (M,K) = 0
. . .то C := C + 1
. . . M := M/K
. . .иначе признак := 'не делится'
. . все
. кц
кон ----------------------------------------
алг цел ОСТ(цел X, У)
 дано
 надо
нач
. знач := X
. нц пока знач >= У
. . знач := знач - У
. кц
кон ----------------------------------------
```


ПОСЛЕ КОНКУРСА

В. ФЕДОТОВ

Из рукописи конкурсного учебника

О межпредметных связях школьной информатики

Предисловие к публикации в журнале

Согласно решению конкурсной комиссии, в журнале «Информатика и образование» публикуются фрагменты моей рукописи школьного учебника ОИВТ. В этом предисловии я хотел бы изложить читателям журнала те принципы, которыми я руководствовался при его написании.

Текст учебника адресован непосредственно школьникам. На составление подробных методических указаний для учителя не хватило времени, да и усло-

От редакции. Представленные фрагменты рукописи отражают взгляд автора на информатику и кибернетику. В них описывается нетрадиционный подход к изложению учебного материала по предмету ОИВТ. Манера, в которой написана рукопись, столь отлична от привычной формы изложения учебного материала в учебниках естественного цикла, что невольно задумываешься: а может быть, прав автор и информатику должны преподавать учителя-гуманитарии?

виями конкурса не было предусмотрено представление подобных материалов. В качестве приложения к рукописи мною была подготовлена небольшая сопроводительная записка, адресованная членам конкурсной комиссии. В нее же были включены и методические установки — слишком краткие для рядового учителя, но вполне достаточные для специалиста. К сожалению, сопроводительная записка не была размножена вместе с рукописью. Практически все замечания рецензентов отпали бы в случае их знакомства с содержанием сопроводительной записки.

Несмотря на то что непосредственной причиной введения ОИВТ в среднее образование явилось бурное развитие микропроцессорной техники, автор в своей рукописи стремился подчеркнуть несводимость информатики только к программированию и показать богатейшие межпредметные связи — не только с математикой и физикой, но также с родным и иностранными языками, биологией, техническим трудом, музыкой. Поэтому вместо краткого введения даны две вводные главы: первая посвящена понятию информации, а вторая — знакомству с ЭВМ (до уровня диалога в режиме «меню»).

Один из моих методических принципов — активная опора на уже имеющиеся знания и опыт учащихся. Поэтому в главе 1 примерами машин по обработке информации служат общеизвестные электробытовые приборы — от утюга до телевизора и магнитофона. Поэтому же в главе 3 основные конструкции алгоритмических языков представлены как аналоги соответствующих средств языков естественных. Эта глава так и называется «Грамматика алгоритмических языков» и занимает центральное место в рукописи.

Учебники А. П. Ершова и других, хотели того авторы или нет, отразили весьма традиционный взгляд на ЭВМ — как большой арифмометр. Львиную долю задач занимают упражнения чисто вычислительного характера, пусть даже по содержанию они происходят из физики, химии или иных дисциплин. Преподаванием ОИВТ заняты программисты, а также учителя математики и физики. В своей рукописи (и особенно в главе 3) я стремился показать прямо противоположный подход: **ближайшими к информатике смежными дисциплинами являются не математика и физика, а языки.** Отказавшись от включения в рукопись вычислительных задач, я попытался сделать учебник, **преподавать по которому смогли бы филологи** — учителя русского языка, иностранных языков, национальных (родных) языков народов СССР.

Острую критику рецензентов, включая секретаря конкурсной комиссии, вызвало выделение систем счисления в отдельную главу из 15 параграфов. Между тем назначение этой главы (4) было четко определено в сопроводительной записке. Для школ, имеющих ВТ, материал этой главы следовало рассматривать как справочный, ограничиваясь только системами счисления, используемыми в имеющейся машине. Для таких школ главу 4 (и частично 3) должен заменять вкладыш со сведениями об алгоритмическом языке конкретной ЭВМ и соответствующим подбором упражнений и задач. В школах же, не имеющих даже микрокалькуляторов, глава 4 несет двойную нагрузку. Наряду с подготовкой к разговору об устройстве ЭВМ и способах хранения и преобразования информации она служит полигоном для отработки навыков составления, записи, формального исполнения алгоритмов. Упражнения на перевод чисел из одной системы счисления в другую или на выполнение арифметических действий хотя и отсутствуют в рукописи, но по аналогии с разобранными в тексте примерами могут быть составлены в неограниченном количестве не только учителем, но и школьником.

Двойную нагрузку несут и остальные главы. Рассказ о физических принципах и устройстве ЭВМ соединен в главе 5 с историческим очерком развития ВТ. Но при этом сама история прокручивается несколько раз, иллюстрируя различные оттенки термина «поколение ЭВМ». Материал же, относящийся к аппаратным и программным средствам, напротив, разделен в две главы — 6 и 7. Это сделано с целью формирования у учащихся более четких представлений

о возможностях ЭВМ, а также для активизации внутрипредметных связей. Наконец, в главе 8 на одном и том же примере иллюстрируется сначала организация перебора вариантов методом обхода дерева, а затем — проблема достоверности результатов машинных вычислений.

Жесткий лимит времени привел к цейтноту, и автор отказался от включения в рукопись недостаточно продуманных вкладышей с задачами, а также рисунков и схем. По той же причине заключительные главы проработаны менее тщательно, чем первые.

Оглавление

Понятие об информации

28 Дорогие юноши и девушки! Вы начинаете изучать основы информатики и вычислительной техники. Этот предмет нов не только для вас: впервые он появился в школе совсем недавно — в 1985 г. Тогда же было организовано отделение информатики Академии наук СССР, а спустя год — образован Государственный комитет СССР по вычислительной технике. Сравнение этих дат показывает, как быстро вычислительная техника вторгается в нашу жизнь, завоевывая одну за другой различные сферы человеческой деятельности.

Слово «информация» наверняка известно всем из вас. Каждый вечер по телевизору передают информационную программу «Время». Слушатели радиопрограммы «Маяк» могут через каждые полчаса услышать информационный выпуск. Разнообразную информацию о политических событиях, достижениях передовиков труда, культурных и спортивных мероприятиях и о многом другом вы можете найти в газетах. За узкоспециальной информацией вы обращаетесь к справочникам, словарям, энциклопедиям и другим книгам, а информация о наличии этих книг в библиотеках содержится в каталогах.

С пониманием слова «информация» вы не испытываете затруднений. Однако попытайтесь дать точное определение! Такого определения вы не найдете даже в пятитомной «Математической энциклопедии», издание которой завершилось в 1985 г. Удивляться этому не нужно: понятие информации относится к числу основ современной науки, наравне с понятиями вещества, энергии и т. п., но если естествознание изучалось столетиями, то кибернетика и информатика значительно моложе. Единого и общепринятого определения информации наука пока не выработала, что не мешает развитию наук об информации.

Обратившись к конкретным примерам, вы сумеете выделить **источник информации** и ее **получателя.** Связь между ними далеко не всегда взаимная, как между отправителем и адресатом почтовой корреспонденции или между абонентами в телефонном разговоре. Чаще всего информация от одного источника расходится ко многим получателям. Случается, что информация попадает не к тому, кому она адресована, либо получатель не знает, от кого получена информация. Однако непременным условием для возможности передать или получить информацию является совместимость «языков» источника и получателя, т. е. способность получателя расшифровать и понять принятую информацию. Слово «язык» взято здесь в кавычки не случайно: это может быть и азбука Морзе, и способ записи информации на магнитную ленту или иное техническое средство, язык жестов или «язык природы».

Прием и передача информации не являются привилегией человека. Различные формы обработки информации мы можем найти как в живой, так и в неживой природе на всех уровнях ее структурной организации. Например, тяготение «сообщает» небесным телам информацию об их взаимном расположении, заставляя их двигаться по вполне определенным траекториям. Недра планеты хранят богатую информацию о различных этапах ее геологической истории. Паводок на реке сохраняет информацию о недавнем ливне. Животный и растительный миры обязаны своим разнообразием механизму передачи наследственной информации с помощью генетического кода.

Во всех этих примерах информация не была адресована человеку, но оказалась доступной для получения и расшиф-

ровки. Наблюдая за светом, за излучениями, исходящими от далеких звезд и галактик, астрономы делают выводы об их строении, движении, развитии. Разгадав тайну наследственности, биологи открыли врачам способы борьбы со многими болезнями. Можно не сомневаться, что информация о целом ряде еще не изученных природных процессов и явлений остается пока недоступной для человека, но со временем может быть извлечена и использована.

Что изучает информатика?

Наивный ответ на вопрос в заголовке этого параграфа, будто информатика изучает информацию, верен лишь отчасти. Информация разнообразна по своим проявлениям, свойствам и приложениям, и изучением ее с различных сторон и точек зрения занимается не одна научная дисциплина, а несколько. Все они в совокупности образуют кибернетику (подобно тому как алгебра, геометрия, математический анализ и некоторые другие дисциплины образуют математику). Итак, информатика, прежде всего, один из разделов кибернетики.

Термин «кибернетика» (буквально означающий «искусство управления») употреблял еще древнегреческий философ Платон. Основоположником кибернетической науки считается американский математик Норберт Винер, определивший ее предмет как управление и связь в животном и машине. Соответственно, в составе кибернетики есть дисциплины, тяготеющие к биологии или технике, занимающиеся преимущественно проблемами управления или связи. Содержанием информатики являются разнообразные аспекты управления, возникающие в связи с использованием всевозможных ЭВМ или, как следовало бы их более точно назвать, машин по обработке информации.

Информатика сформировалась к середине 60-х г г. XX в. Истоки ее находились на стыке библиотечного дела, лингвистики и счетной техники. Первоначально основным содержанием информатики было изучение закономерностей накопления информации и методов ее поиска, а едва ли не единственным полем приложения — библиотечные каталоги.

Последующее бурное развитие вычислительной техники привело к изменению содержания информатики. Библиотечные мотивы постепенно отходили на второй план. Уже к середине 70-х гг. создавались мощные информационно-поисковые и информационно-справочные системы разнообразного назначения с использованием ЭВМ и магнитных носителей большой емкости. Информатику стали понимать как дисциплину, изучающую структуру и общие свойства информации, закономерности ее создания, преобразования, передачи, хранения и использования в различных сферах человеческой деятельности.

Серьезное внимание на содержание информатики оказал информационный взрыв второй половины XX в. Оказалось, что легче, дешевле и быстрее заново поставить дорогостоящий научный эксперимент, чем найти статью с его описанием, даже в том случае, когда заранее известно о том, что подобный эксперимент был поставлен. Более же типична другая ситуация: один и тот же предмет или природное явление изучается узкими направлениями несмежных наук, описывается в совершенно различных терминах, а сами исследователи даже не подозревают о том, что их работа кем-то уже проделана. Случается и наоборот: одни и те же термины используются разными науками в совершенно никак не связанных между собой смыслах (как, например, «тело», «функция», «операция», «инъекция» и многие другие — в математике и медицине), что серьезно затрудняет использование алфавитных и предметных каталогов в библиотеках, поиск информации по ключевым словам. Возникла проблема классификации накопленных знаний, извлечения содержащейся в них информации, организации разветвленных информационных сетей, баз и банков данных сложной структуры.

И вновь лицо информатики изменилось благодаря прогрессу в области вычислительной техники. Совершенствование элементной базы привело с середины 80-х г г. к массовому производству и ис-

пользованию персональных компьютеров. Вычисления перестали быть основной областью применения ЭВМ, так как их вытеснили различные виды обработки текстовой и графической информации. Зародившаяся как раздел кибернетики, информатика вышла за рамки узкого научного направления и сама стала одной из фундаментальных наук широкого профиля с многочисленными собственными подразделами.

Научить вас ориентироваться в необъятном море информации, суметь эффективно отобрать нужную для вас информацию и оценить ее достоверность — одна из задач школьного предмета ОИВТ.

Основные устройства ЭВМ*

По своему строению вычислительная машина аналогична живому существу. Организм животного включает мозг, систему кровообращения, органы чувств, органы речи. Сходные им по функциональному назначению устройства можно найти и в составе ЭВМ. Правда, в отличие от животных, ЭВМ не способна самостоятельно передвигаться. Важнейшей составной частью — «мозгом ЭВМ» — является процессор. Аналогом органов кровообращения служит система электропитания. Роль органов чувств играют разнообразные устройства ввода, а органов речи — устройства вывода информации.

Для хранения промежуточной информации в составе ЭВМ предусмотрены разнообразные запоминающие устройства. Такие устройства могут входить в состав процессора (тогда они называются регистрами), быть непосредственно связаны с процессором (оперативные запоминающие устройства или ОЗУ) либо быть связаны с процессором через специальные устройства ввода-вывода (долговременные запоминающие устройства или ДЗУ). Выделяют также постоянные запоминающие устройства (ПЗУ), в которых постоянство и повышенная надежность хранимой информации обеспечиваются за счет иной технологии их изготовления. Если в роли ОЗУ выступают те разделы мозга, которые обеспечивают память человека, т. е. процессы запоминания и вспоминания, то роль процессора выполняют разделы мозга, управляющие работой сердца, легких, других жизненно важных органов и обеспечивающие собственно мыслительную деятельность, а аналогом регистров служат те клетки мозга, в которых фиксируется информация, непосредственно поступающая в мозг в виде слабых электрических сигналов (!) от органов чувств. В качестве же ДЗУ человек может использовать записные книжки, справочники (ПЗУ) и... ДЗУ вычислительных машин.

Процессоры современных ЭВМ имеют размеры порядка нескольких сантиметров и даже миллиметров. Поэтому нередко даже специалисты, работающие на таких машинах, затрудняются ответить на вопрос: где же именно находится процессор? Чаще всего он расположен в одном корпусе с клавиатурой (основное устройство ввода), но может быть и в составе дисплея (сходное с телевизором устройство вывода, на экране которого визуально проверяется вводимая с клавиатуры информация и куда выводятся основные сообщения о результатах работы процессора). Миниатюрные размеры процессора позволяют разместить его даже внутри шнура, соединяющего клавиатуру с дисплеем.

Как уже сказано, основным устройством ввода является клавиатура. Иногда ее называют пультом управления, однако в применении к современным ЭВМ такое название считается устаревшим. Клавиатура ЭВМ гораздо богаче, чем клавиатура микрокалькулятора или пишущей машинки. На ней вы можете увидеть буквы разных алфавитов, цифры, знаки препинания и арифметических операций, специальные математические или иные символы (в зависимости от назначения машины). Есть на клавиатуре различные управляющие клавиши — для смены регистров (что позволяет совмещать на основных клавишах несколько разных символов, выбирать алфавит, заглавные или строчные буквы), для управления размещением текста на экране дисплея и в программной па-

* Звездочка означает, что содержание параграфа публикуется с сокращениями.— *Примеч. ред.*

мяти, для непосредственного управления работой процессора. Нажатие на некоторые клавиши приводит к набору не отдельных символов, а целых слов и предложений (в частности, основных управляющих инструкций), причем выбор этих наиболее употребительных текстов вы можете изменить по своему усмотрению.

ЭВМ обрабатывает информацию гораздо быстрее человека. Поэтому ввод с клавиатуры очень длинных текстов, программ, числовых данных и других массивов информации приводит к неоправданным простоям дорогостоящей техники. С целью более рационального использования ЭВМ применяются другие устройства ввода. Ко многим ЭВМ подсоединены десятки терминалов, включающих клавиатуру, дисплей и запоминающее устройство. На таком терминале можно подготавливать информацию и тогда, когда ЭВМ выключена. Когда же вся вводимая информация будет подготовлена и проверена, она прямо с терминала посылается в ЭВМ. Если же ЭВМ в этот момент еще выключена или занята решением задачи, поступившей с другого терминала, то для долговременного хранения информации ее можно записать на магнитную ленту или диск (для чего к терминалу может быть подключен обычный магнитофон или специально предназначенный для этого дисковод).

На терминал же поступят и сообщения с ЭВМ о ходе и результатах решения задачи. Поэтому терминал является устройством не только ввода, но и вывода информации. В состав терминала может входить и процессор, хотя и значительно менее мощный, чем в основной ЭВМ. В этом случае терминал можно рассматривать как самостоятельную микро-ЭВМ, включенную в состав вычислительного комплекса. Крупные вычислительные комплексы включают не только систему терминалов с самостоятельными процессорами, но и несколько ЭВМ (как правило, различных по своим возможностям и специализации). В школах, где вычислительная техника используется только для учебных целей и не приходится обрабатывать большие объемы информации, нередко устанавливаются комплекты учебной вычислительной техники (КУВТ), включающие только терминалы с микропроцессорами, но без центральной ЭВМ либо с точно таким же микропроцессором на терминале учителя.

ЭВМ может выдавать и звуковую информацию. В большинстве случаев подобные устройства вывода примитивны и служат только для привлечения внимания оператора к возможным сбоям в работе машины (конец или обрыв бумаги, отсутствие или отказ нужного устройства и т. п.). Однако возможности языков программирования позволяют синтезировать музыку и даже речь, для звучания которых к ЭВМ в качестве устройства вывода могут подключаться громкоговорители или те же магнитофоны.

В состав многих ЭВМ входят аналого-цифровые преобразователи, позволяющие вводить в ЭВМ информацию непосредственно с электроизмерительных приборов различного назначения (физических, медицинских, метеорологических, технической диагностики и др.), а также информацию, поступающую по различным радио- и телевизионным каналам (в том числе с морских судов, самолетов, космических аппаратов) и линиям связи (телефонной, телеграфной, телетайпной). Графическая информация может с помощью специального светового пера вводиться прямо через дисплей. Существуют и устройства ввода, выполняющие функции зрения (наибольшее распространение они получили в полиграфии — фотонабор).

Столь же разнообразны и устройства вывода, имеющиеся в специализированных ЭВМ. Минуя человека, выводимая информация может приводить в движение различные механизмы, управлять сложнейшими технологическими процессами.

Наконец, нужно назвать еще и имеющиеся во всех ЭВМ каналы передачи информации, по которым осуществляется связь между различными устройствами. Чаще всего это обычные провода, соединенные в толстые шнуры. Но нередко эта связь осуществляется и без проводов — как в радио и телевидении.

Программа «Тетраэдр»

Потехе — час, время — делу. Познакомимся теперь с одной из учебных программ. Из их списка выбираем «Тетраэдр» и, нажав клавишу, отвечающую номеру этой программы в меню, вызываем ее. На экране появляется пояснительный текст, из которого мы узнаем, что программа решает геометрическую задачу: вычисляет объем тетраэдра (треугольной пирамиды), площадь его поверхности и периметр (сумму длин всех ребер).

Задача непростая. Если для нахождения периметра достаточно всего лишь сложить 6 чисел — длины 6 ребер тетраэдра, то нахождение площади поверхности требует четырехкратного применения формулы Герона, а объем тетраэдра выражается через длины его ребер по такой сложной формуле, что она даже не изучается в школьном курсе.

Дисплей «сообщает», что длины ребер тетраэдра вы должны выбрать сами, затем ввести в машину с клавиатуры. Для определенности, сообщает текст на экране, длины ребер обозначены первыми шестью буквами латинского алфавита, причем первые три образуют треугольник одной из граней, а три последующих соответственно попарно скрещиваются с ними. Подсказка «A?» и яркий (на некоторых дисплеях — мигающий) квадратик после нее (он называется курсором) указывают, что машина ожидает ввода числа A — длины первого ребра.

Чтобы не долго думать, нажмем клавишу «1». Но почему машина долго не реагирует, а курсор остался в той же строке, сразу вслед за единицей? Да так и должно быть: машина ведь «не знает», сколько цифр во вводимом числе, и ожидает ввода последующих цифр. Как сообщить ей, что ввод числа закончен? По-видимому, большая клавиша с надписью «ВВОД» служит именно для этого. Нажатие на нее подтверждает наше предположение — курсор исчез, а затем в следующей строке появилась новая подсказка «B?», приглашающая к вводу длины второго ребра.

Попробуем набрать в качестве длин ребер первое, что пришло в голову,— натуральные числа от 1 до 6. После ввода длины шестого ребра курсор больше не появляется, но тотчас же на очередной строке экрана появилось сообщение: «Проверьте правильность ввода. Длины ребер: 1, 2, 3, 4, 5, 6. В случае ошибки, нажмите клавиши «ПРЕРЫВАНИЕ» и «ПУСК», а затем повторите ввод заново». Однако не успели мы не только проверить ввод, но даже прочитать это сообщение, как на экране появилось новое: «Ребра A, B и C не образуют треугольника. Повторите ввод данных», после чего вновь возникла подсказка «A?», предлагающая опять начать с длины первого ребра.

На этот раз поспешность привела нас к ошибке не в обращении с машиной, а в геометрии. Действительно, числа A=1, B=2 и C=3 не могут быть длинами сторон треугольника, так как 1+2=3, а сумма любых двух сторон треугольника всегда больше третьей стороны.

Будем более осторожны в выборе данных и, прежде чем вводить их, сами проверим выполнение всех неравенств. В качестве длин первых трех ребер введем «9», а затем три раза — «5». Но опять неудача! «Треугольники не образуют тетраэдра»,— сообщает дисплей. Действительно, равнобедренные треугольники со сторонами 5, 5 и 9 имеют тупой угол в вершине и сравнительно малую высоту, поэтому три таких треугольника не покрывают равностороннего треугольника со сторонами 9, 9 и 9 и не могут вместе с ним служить гранями одного тетраэдра.

Площади граней тетраэдра должны удовлетворять неравенствам, аналогичным неравенствам треугольника: площадь каждой грани не может быть больше суммы трех остальных граней или быть ей равной. Именно это условие и было только что нарушено. Чтобы наверняка удовлетворить и ему, возьмем в качестве граней одинаковые треугольники.

Попытаемся взять тетраэдр, все грани которого — египетские треугольники. Для этого последовательно вводим в машины длины ребер: «3», «4», «5», «3», «4», «5». Однако неудача ждет нас и на этот раз: тетраэдр выродился в дважды

покрытый прямоугольник. Оказывается, далеко не любые 6 чисел могут служить длинами ребер тетраэдра и соотношения, которым они должны при этом удовлетворять, вовсе не так очевидны, как может показаться на первый взгляд. Не обращаясь к компьютеру, мы могли бы убедиться в этом, только затратив несравненно больше времени.

После стольких неудач будем действовать наверняка: приравняем все длины к единице. Разумеется, на этот раз тетраэдр существует, и на экране мы читаем его размеры: «Объем= =0.11785112, площадь=1.7320508, периметр=6».

Структура оператора*

Как вы знаете из грамматики родного или иностранного языка, любое предложение на естественном языке содержит главные (подлежащее и сказуемое) и второстепенные (определения, обстоятельства и др.) члены. В конце каждой фразы ставится знак препинания — чаще всего точка, но, возможно, восклицательный или вопросительный знак.

Примерно такова же структура и оператора — предложения на алгоритмическом языке. Есть, однако, и некоторые различия. Из двух главных членов в предложении алгоритмического языка обычно присутствует только один, а нередко опущен и он. Действительно, чаще всего оператор представляет собой приказ, т. е. повелительное предложение. Подлежащим в нем является местоимение второго лица («ты» или «вы»), и обычно оно опускается. Если же вместо повелительного наклонения приказ передан неопределенной формой глагола, то подлежащее отсутствует изначально — предложение безличное.

Другой тип инструкций — описания (определения), содержащие потенциальный приказ или предварительные пояснения. В таких предложениях есть содержательное подлежащее, но сказуемое выражается глаголом «быть» в форме третьего лица единственного числа — «есть» — и, как правило, опускается. Нередко этот глагол скрыт внутри знака равенства.

Далее заметим, что точка почти никогда не используется в алгоритмических языках в качестве знака конца предложения. Дело в том, что в англоязычной литературе (а вслед за ней и почти во всех языках программирования) точка используется для разделения целой и дробной частей смешанного десятичного числа (в отличие от русской литературы, не посвященной вычислительной технике и программированию, где для этой же цели обычно используется запятая).

К сожалению, в разных алгоритмических языках конец оператора обозначается разными способами. Например, в Алголе для этой цели служит точка с запятой, в Бейсике — двоеточие, в Коболе — точка, в Фортране — символ конца строки, поэтому, начиная изучение какого-либо алгоритмического языка, вы должны первым делом узнать и твердо запомнить, какой символ служит в этом языке для обозначения конца оператора.

Основные глаголы*

Обычно языки программирования используют сравнительно небольшое число стандартных глаголов. Это связано с тем, что каждый глагол обозначает то или иное действие исполнителя, а круг таких простейших действий ограничен в конечном счете системой команд. Кроме того, нередко вместо глагола приказ отдается в форме существительного: ВВОД (вместо ВВЕДИ), ВЫВОД (вместо ВЫВЕДИ), ЧТЕНИЕ (вместо ЧИТАЙ), ЗАПИСЬ (вместо ПИШИ), ПЕЧАТЬ (вместо ПЕЧАТАЙ) и т. п. Хотя в естественных языках эти слова и являются существительными, будем считать их глаголами алгоритмического языка, в соответствии с их фактической ролью. Строго следуя этому соглашению, считаем глаголом и вопросительный знак, играющий во многих языках ту же роль, что и инструкция ВЫВОД или ПЕЧАТЬ.

Роль глаголов играют также знаки различных операций (равенства, арифметические, обозначения функций и др.).

Заметим сразу же, что сами обозначения знаков действий в алгоритмических языках несколько отличаются от используемых в математике. Без изменения остаются только знаки сложения («+» — плюс) и вычитания («—» — минус).

Как уже сказано, ряд глаголов (или выполняющих их роль существительных, а также знак вопроса) относится к операциям ввода-вывода. В принципе, здесь хватило бы только двух или даже одного служебного слова, а конкретное устройство, направление обмена и формат данных можно указывать в списке аргументов этой операции. Однако, по традиции, за некоторыми основными устройствами ввода-вывода (клавиатура, экран, принтер, магнитофон, дисковод и др.) обычно закреплены индивидуальные названия соответствующих операций обмена.

Есть слова, которые относятся к непосредственному управлению программой. Это, прежде всего, ПУСК (в некоторых языках — БЕГИ или иностранный вариант этого слова) — для запуска программы на исполнение, ЛИСТ — для ее чтения (т. е. вывода на экран или принтер), а также УДАЛИТЬ, ПРИСОЕДИНИТЬ и др.

Сравнительная малость числа основных глаголов компенсируется во многих языках возможностью выводить по своему усмотрению новые действия — процедуры. Подробнее с правилами их описания и использования мы познакомимся позднее. *Окончание следует*

ЧТО? МОЖЕТ БЫТЬ?

Играть или учиться играючи?

Вас никогда не душили спруты? А в темном углу лабиринта не проглатывал монстр, стерегущий свои, хотя и существующие только на экране компьютера, но такие желанные богатства? Скучно?

Не увлекаются теперь такими играми и студенты многих американских колледжей и университетов. Но не потому, что у них нет компьютеров или остыл игровой азарт. Просто в их руки попала намного более увлекательная обучающая программа «Chemaze», позволяющая не только отлично провести время, но и быстро освоить основные законы неорганической химии.

В этой обучающей программе вы не встретите каких-либо поучений или инструкций. Делайте что хотите, программа лишь точно и ехидно прокомментирует то, что с вами произошло. А случиться в мире химических реактивов может самое неожиданное. Особенно если времени на раздумье остается мало.

Идея игры очень проста: по всему пространству экрана, разбитому на множество связанных проходами участков, расставлены сосуды с различными реагентами. Ловко лавируя между ними, вы должны провести вашу колбу, например, с соляной кислотой в заданное место, не пострадав при этом от происходящих с вами по дороге слишком бурных химических реакций. Долго выбирать дорогу вам не даст энергичная кювета с потенциально опасным для вашей колбы содержимым. Ее задача — догнать вас и прореагировать с тем составом, который в данный момент находится в вашей колбе.

Программа ведет счет ваших больших и маленьких успехов и неудач, который лучше всякого экзамена демонстрирует ваш уровень знания химии и умение быстро аналитически мыслить. Но даже если вы только начинаете изучать эту сложную и интересную науку, не бойтесь: ни один даже самый большой взрыв не причинит вам никакого вреда. Быть может, это придаст вам смелости в изучении самых невероятных химических превращений.

Весь пакет обучающих программ «Chemaze» умещается на десяти флоппи-дисках и помимо цветной графики, мультипликации и моделирования экспериментов включает справочник по химическим элементам, каталог химических реактивов, измерительную систему и игру в химические реакции. Программы легко модифицируются, и их сложность легко может быть изменена в зависимости от уровня подготовленности учащихся.

Компьютер и автосервис

Американский репортер Р. Сикорский проверил качество автосервиса в своей стране таким способом — проехал по 33 штатам на автомобиле, заезжая на станции автообслуживания. Перед каждым визитом он отсоединял высоковольтный провод от одной из свеч зажигания. Результат путешествия таков — простейшая неисправность была правильно найдена и устранена лишь в 28 % случаев.

Пытался определить, что случилось, и компьютер. Целый час механики обследовали с его помощью автомобиль репортера-следователя, и в конце концов электронный диагност решил, что нужно заменить крышку трамблера и бегунок. Довольно близко к истинной неисправности, но не более того.

Возможно, создатели диагностической системы не предполагали, что высоковольтный провод может быть просто отключен, ну а дальше... Компьютер — это мельница: засыплешь зерно — выдаст муку, засыплешь мусор — выдаст мусор.

8 декабря 1988 г. после продолжительной и тяжелой болезни скончался ведущий советский ученый, основоположник школьной информатики, член редколлегии журнала «Информатика и образование» академик Андрей Петрович Ершов.

Научная биография А. П. Ершова неотделима от становления системного и теоретического программирования, развития этих направлений, формирования их методологической базы. Ему принадлежат фундаментальные результаты в теории операторных схем, общей теории экономии памяти, в теории схем программ с распределенной памятью, а также начальная формулировка теории схем параллельных программ, развитая впоследствии его учениками.

А. П. Ершов — автор одной из первых в мире Программирующих программ, той концептуальной основы, на которой строятся современные алгоритмические языки и системы программирования. Им создан Входной язык — один из первых алгоритмических языков высокого уровня. Книга А. П. Ершова «Программирующая программа для быстродействующей электронной счетной машины» — первая в мировой практике монография по автоматизации программирования.

Под руководством А. П. Ершова созданы система «Альфа», решившая проблему разработки высококачественных рабочих программ средствами автоматизации программирования, а также первая в СССР система разделения времени «Аист», архитектурные и программные решения которой широко применяются при создании вычислительных центров коллективного пользования.

В последние годы теоретические работы А. П. Ершова были связаны с выявлением фундаментальных основ трансляции, развитием теории смешанных вычислений и трансформационного подхода к программированию. Его работы по теории смешанных вычислений были первым исследованием по программированию, удостоенным высшей в СССР математической награды — премии им. А. Н. Крылова.

А. П. Ершов придавал большое значение организационной работе. Он был одним из инициаторов создания и активным членом Отделения информатики и вычислительной техники АН СССР, возглавлял комиссию по системному математическому обеспечению Координационного комитета по вычислительной технике АН СССР, был главным редактором журнала «Микропроцессорные средства и системы», председателем научного совета АН СССР по комплексной проблеме «Кибернетика».

Андрей Петрович пользовался высоким научным авторитетом не только в нашей стране, но и за рубежом. Он удостоен награды МФОИ (Международной федерации по обработке информации) — «Серебряного сердечника», был почетным членом Британского общества вычислительной техники, членом редколлегий четырех зарубежных журналов по компьютерной тематике.

Обладая широким научным кругозором и умением видеть перспективу, А. П. Ершов одним из первых в нашей стране понял стратегическое значение информатики в научно-техническом и социальном развитии.

В середине 70-х гг. А. П. Ершов начинает не всеми понятую тогда работу по ознакомлению школьников с миром вычислительной техники, по научному и экспериментальному обоснованию идей информатизации образования. Будучи признанным лидером в этой области, он много сделал для становления нового общеобразовательного учебного предмета, для появления отечественных школьных ЭВМ.

А. П. Ершов награжден тремя орденами Трудового Красного Знамени, орденом «Знак Почета», медалями СССР, был лауреатом Государственной премии СССР.

Светлая память об Андрее Петровиче Ершове, ученом и человеке, навсегда сохранится в наших сердцах.

ТЕЛЕГРАММА

Методисты кабинетов информатики всех институтов усовершенствования учителей РСФСР, находящиеся на курсах в Москве, скорбят о тяжелейшей утрате — кончине основоположника советской школьной информатики академика Андрея Петровича Ершова. Светлую память о нем мы навсегда сохраним в своих сердцах.

109 подписей

КООПЕРАТИВ "ИНФОРМ"

ПОСТАВЛЯЕТ

систему программированного обучения "Рига"
для КУВТ-86 (БЭЙСИК и ФОКАЛ)
с методическими материалами и (SIC!)
обеспечивает сопровождение системы у заказчиков.
Система "Рига" - это программное средство обучения
разнообразным общеобразовательным предметам,
постоянная обратная связь учащихся с преподавателем,
база знаний на машинных носителях,
"электронный журнал".
Т-язык системы "Рига" позволяет легко
программировать новые уроки, задачники, тесты,
конспекты в форме увлекательных игр с графическим
оформлением и музыкальным сопровождением.

Кооператив "Информ" также поставляет для микро-ЭВМ "Электроника БК-0010" ряд программ, которые представляют собой технологический набор "деталей", "узлов" и "инструментов".
Набор дает возможность создавать даже на компьютере, имеющем только интерпретатор ФОКАЛа,
мощные программные системы для образования,
культурного досуга, расчетов, математического
моделирования, для автоматизации исследований
и учебного процесса.
Технологические программы: система ассемблера МИКРО, программа-отладчик ГРОТ, перемещаемый Интерфейс Внешних Функций интерпретатора ФОКАЛ-БК0010 с генератором систем внешних функций и библиотекой внешних функций, другие программы и системы.
Все программы поставляются с руководствами.

Переписка:
142292, Московская обл., г. Пущино,
микрорайон "В", д.29, кв.32, "ИНФОРМ"

Телефоны (московские):
497-86-52 (с 19-00 до 21-00), Л.Н.Майоров:
923-74-67 доб.2-93 (с 12-00 до 13-00),
923-96-68 доб.2-93 (с 12-00 до 13-00),
А.П.Казанцев

А. АРХАНГЕЛЬСКИЙ

Мир ЭВМ

8. Продолжим наши игры

8.1. Устройство виртуальной памяти

Воображение — в действительности не что иное, как вид памяти, освобожденный от уз времени и пространства.

С. Т. Колридж

«Суть концепции виртуальной памяти заключается в том, что адреса, к которым обращается выполняющийся процесс, отделяются от адресов, реально существующих в первичной памяти» — такое определение виртуальной памяти дает в своей книге Х. Дейтел. Но как же работать с устройством, которое не является реальным объектом?

Да, виртуальная память (напомним, что она появилась в версии 5.0) не существует как некоторое самостоятельное устройство. Вспомним, однако, что мы говорили о файловой системе; файловая система может быть организована в любой памяти — так почему бы и не в ОЗУ? В «сотке» объем ОЗУ — 356К байт, т. е. такой же, как объем гибкого диска, а монитор RT11SJ вместе с периферийными устройствами и программами пользователя использует только 64К байт. Попробуем в оставшуюся память записать необходимую информацию (о которой мы говорили в пятой главе), а после этого работать с ней как с устройством, имеющим файловую систему с произвольным доступом. Для начала, видимо, нужно инициализировать эту файловую систему*.

```
.INITIALIZE VM0:⟨CR⟩
VM0:/INITIALIZE — ARE YOU SURE?
Y⟨CR⟩
```

Ничего страшного не произошло. Система молча поставила точку. Сразу можно отметить, что так как память у процессора одна, то номер устройства значения не имеет, он просто игнорируется.

Теперь можно посмотреть каталог устройства VM:

```
.DIR/FU/BL/VOL VM:⟨CR⟩
14-MAR—88
VOLUME ID: RT11A
OWNER:
⟨UNUSED⟩ 376          8
0 FILES, 0 BLOCKS
376 FREE BLOCKS
```

Будем копировать сюда файлы, которые имеются на нашем диске. Сделать это можно быстро. Не забывая про системные файлы, даем групповую (довольно громко сказано) спецификацию, введя команду

```
COPY/SYSTEM SY: VM:⟨CR⟩
    FILE COPIED:
```

Продолжение. Начало см.: Информатика и образование. 1987. № 1—3; 1988. № 3—6.

* Напоминаем, что курсивом выделяются приказы, набираемые человеком, а ⟨CR⟩ — обозначение клавиши ввода.— *Примеч. ред.*

SY:SWAP .SYS TO VM:SWAP .SYS
SY:RT11SJ. SYS TO VM:RT11SJ .SYS
SY:TT .SYS TO VM:TT .SYS
SY:LD .SYS TO VM:LD .SYS
SY:DX .SYS TO VM:DX .SYS
SY:VM .SYS TO VM:VM .SYS
SY:RK .SYS TO VM:RK .SYS
SY:DP .SYS TO VM:DP . SYS
SY:MT .SYS TO VM:MT . SYS
SY:LP .SYS TO VM:LP . SYS
SY:PC .SYS TO VM:PC . SYS
SY:NL .SYS TO VM:NL .SYS
SY:SL .SYS TO VM:SL .SYS
SY:PIP .SAV TO VM:PIP .SAV
SY:DUP .SAV TO VM.DUP .SAV
SY:DIR .SAV TO VM:DIR .SAV
SY:LET .SAV TO VM:LET .SAV
SY:K52 .SAV TO VM:K52 .SAV
SY:DUMP .SAV TO VM:DUMP .SAV
SY:EDIT .SAV TO VM:EDIT .SAV
SY:FORMAT .SAV TO VM:FORMAT .SAV
SY:PATCH .SAV TO VM:PATCH .SAV
SY:STARTS .COM TO VM:STARTS .COM
SY:V5NOTE .TXT TO VM:V5NOTE .TXT
?PIP—F—DEVICE FULL VM:IND .SAV

— Стоит ли комментировать копирование каждого файла? — могут спросить читатели.— Ведь часто мы знаем, что копируем. Нельзя ли запретить программе этот многословный рассказ?

Конечно, мы найдем такую возможность, но сначала отметим, что не все нужные нам файлы скопировались. Пользуясь тем, что эта система у нас создается для эксперимента, удалим некоторые файлы и докопируем то, что нам нужно. Некоторые команды будут подробно обсуждаться дальше, поэтому используем их «молча».

Для начала посмотрим, что есть на устройстве VM:.

DIR/FU/BL/VOL VM:⟨CR⟩

```
 14-MAR-88
 VOLUME ID: RT11A
 OWNER    :
SWAP   .SYS  26P 13-JAN-84   8      RT11SJ.SYS  73P 13-JAN-84  34
TT     .SYS   2P 13-JAN-84 107      LD     .SYS   8P 13-JAN-84 109
DX     .SYS   4P 13-JAN-84 117      VM     .SYS   3P 13-JAN-84 121
RK     .SYS   3P 13-JAN-84 124      DP     .SYS   3P 13-JAN-84 127
MT     .SYS   9P 13-JAN-84 130      LP     .SYS   2P 13-JAN-84 139
PC     .SYS   2P 13-JAN-84 141      NL     .SYS   2P 13-JAN-84 143
SL     .SYS  13P 13-JAN-84 145      PIP    .SAV  29P 13-JAN-84 158
DUP    .SAV  45P 13-JAN-84 187      DIR    .SAV  19P 13-JAN-84 232
LET    .SAV   5P 13-JAN-84 251      K52    .SAV  55P 13-JAN-84 256
DUMP   .SAV   8P 13-JAN-84 311      EDIT   .SAV  19P 13-JAN-84 319
FORMAT.SAV  21P 13-JAN-84 338      PATCH  .SAV   9P 13-JAN-84 359
STARTS.COM   1P 13-JAN-84 368      V5NOTE.TXT   2P 13-JAN-84 369
⟨ UNUSED ⟩  13              371
 24 FILES, 363 BLOCKS
 13 FREE BLOCKS
```

Мы видим, что у нас только 13 свободных блоков, для IND.SAV нужно 52 блока. Для начала удалим два файла — K52.SAV и FORMAT.SAV, используя для этого команду

.DELETE VM:(K52,FORMAT).SAV⟨CR⟩

?PIP-W—PROTECTED FILE VM:K52.SAV
?PIP—W—PROTECTED FILE
VM:FORMAT.SAV

Вот она, сила защиты файлов. Воспользуемся тем, что мы имеем право снять защиту.

.UNPROTECT VM:(K52,FORMAT).SAV⟨CR⟩

Теперь можно и удалить

.DELETE VM:(K52,FORMAT).SAV⟨CR⟩

а затем снова посмотреть каталог

.DIR/FU/BL/VOL VM:⟨CR⟩

```
 14-MAR-88
 VOLUME ID: RT11A
 OWNER    :
SWAP   .SYS  26P 13-JAN-84   8      RT11SJ.SYS  73P 13-JAN-84  34
TT     .SYS   2P 13-JAN-84 107      LD     .SYS   8P 13-JAN-84 109
DX     .SYS   4P 13-JAN-84 117      VM     .SYS   3P 13-JAN-84 121
RK     .SYS   3P 13-JAN-84 124      DP     .SYS   3P 13-JAN-84 127
MT     .SYS   9P 13-JAN-84 130      LP     .SYS   2P 13-JAN-84 139
PC     .SYS   2P 13-JAN-84 141      NL     .SYS   2P 13-JAN-84 143
SL     .SYS  13P 13-JAN-84 145      PIP    .SAV  29P 13-JAN-84 158
DUP    .SAV  45P 13-JAN-84 187      DIR    .SAV  19P 13-JAN-84 232
LET    .SAV   5P 13-JAN-84 251      ⟨ UNUSED ⟩  55              256
DUMP   .SAV   8P 13-JAN-84 311      EDIT   .SAV  19P 13-JAN-84 319
⟨ UNUSED ⟩  21              338      PATCH  .SAV   9P 13-JAN-84 359
STARTS.COM   1P 13-JAN-84 368      V5NOTE.TXT   2P 13-JAN-84 369
⟨ UNUSED ⟩  13              371
 24 FILES, 363 BLOCKS
 13 FREE BLOCKS
```

Пустые места на диске, разбросанные как попало, использовать можно, если они достаточно большие. Но лучше, чтобы свободное место было одним большим куском, а файлы при этом были «прижаты» к началу тома (вспомните — у дисков более надежная запись на внешних дорожках). Найдем в русско-английском словаре слово *сжимать* и используем его в качестве команды.

.SQUEEZE VM:⟨CR⟩

VM:/SQUEEZE — ARE YOU SURE? *Y⟨CR⟩*

Опять «Шура»! Почему система задала этот вопрос? Мы ведь ничего не хотим удалять!

И все-таки вопрос не лишний. При сжатии тома существуют секунды, когда нарушается адекватное соответствие между файлом и записью о нем в каталоге (см. первую главу); кроме того, возможно и нарушение целостности самого файла (переписывание на новое место происходит поблочно). Поэтому система предлагает подумать, уверены ли мы в надежности аппаратуры и носителей.

Впрочем, в последних версиях системы программа DUP стала более умной. Она не только оставляет на месте файлы с типом .BAD, но и может создать их

сама, если обнаружит плохие блоки во время сжатия.

Мы и сами можем принять меры к сохранности информации, указав ключ /OUTPUT:DEV:. В качестве аргумента нужно поставить имя выходного устройства, на которое и будет записан сжатый вариант. При этом в операции не участвуют плохие файлы и загрузчики в начале тома.

Без помощи KMON сжатие делается так:

.RUN SY:DUR⟨CR⟩
**[OUTDEV:=]VM:/S⟨CR⟩*

Отметим, что при нужде можно использовать ключ /WAIT и что при сжатии системного тома происходит автоматическая перезагрузка системы для восстановления указателей на драйверы в таблицах монитора.

Посмотрим на плоды своих трудов.

.DIR/FU/BL/VOL VM:⟨CR⟩

```
 14-MAR-88
 VOLUME ID: RT11A
 OWNER    :
SWAP   .SYS   26P 13-JAN-84   8      RT11SJ.SYS  73P 13-JAN-84  34
TT     .SYS    2P 13-JAN-84 107      LD     .SYS   8P 13-JAN-84 109
DX     .SYS    4P 13-JAN-84 117      VM     .SYS   3P 13-JAN-84 121
RK     .SYS    3P 13-JAN-84 124      DP     .SYS   3P 13-JAN-84 127
MT     .SYS    9P 13-JAN-84 130      LP     .SYS   2P 13-JAN-84 139
PC     .SYS    2P 13-JAN-84 141      NL     .SYS   2P 13-JAN-84 143
SL     .SYS   13P 13-JAN-84 145      PIP    .SAV  29P 13-JAN-84 158
DUP    .SAV   45P 13-JAN-84 187      DIR    .SAV  19P 13-JAN-84 232
LET    .SAV    5P 13-JAN-84 251      DUMP   .SAV   8P 13-JAN-84 256
EDIT   .SAV   19P 13-JAN-84 264      PATCH  .SAV   7P 13-JAN-84 283
STARTS.COM    1P 13-JAN-84 292      VSNOTE.TXT    2P 13-JAN-84 293
( UNUSED )   89              295
 22 FILES, 287 BLOCKS
 89 FREE BLOCKS
```

Скопируем с системного тома недостающие файлы.

.COPY SY:(IND,DATIME) VM:⟨CR⟩

```
      FILES COPIED
SY:IND    .SAV TO VM:IND    .SAV
SY:DATIME.COM  TO  VM:DATIME.COM
```

Вернемся к вопросу об отказе от выдачи системой на экран протокола своих действий. Запретить протокол просто — достаточно использовать ключ /NOLOG, происходящий от слова log, одним из значений которого является «запись о событии в вахтенном или бортовом журнале». Однако следует помнить, что появление очередного сообщения в протоколе копирования точно соответствует совершенно очередного действия и в случае какого-либо сбоя протокол может оказать немалую помощь.

Ну а теперь — что же нужно сделать с устройством VM: (виртуальной памятью), чтобы загрузиться с него как с диска? Для одинаковых функций не нужно придумывать новые команды. Мы рассматриваем устройство VM: как диск, поэтому скопируем загрузчик как на диск и обычным образом загрузим систему.

.COPY/BOOT VM:RT11SJ VM:⟨CR⟩
.BOOT VM:⟨CR⟩

```
  RT-11SJ  V5.1 (S)
.SET TT QUIET

                    RT-11 V5.1
                    ----------

   INSTALLATION OF RT-11 VERSION 5.1 IS COMPLETE AND YOU ARE NOW
EXECUTING FROM THE WORKING VOLUME    (PROVIDED YOU HAVE USED THE
AUTOMATIC INSTALLATION PROCEDURE). DIGITAL RECOMMENDS YOU VERIFY
THE CORRECT OPERATION OF YOUR SYSTEM'S SOFTWARE USING THE
VERIFICATION PROCEDURE.  TO DO THIS, ENTER THE COMMAND:

                        IND VERIFY

     NOTE THAT VERIFY SHOULD BE PERFORMED ONLY AFTER THE DISTRI-
BUTION MEDIA HAVE BEEN BACKED UP.  THIS WAS ACCOMPLISHED AS PART
OF AUTOMATIC INSTALLATION ON  ALL  RL02,  RLX02  AND  RX50 BASED
SYSTEMS,    INCLUDING THE MICRO/PDP-11 AND THE PROFESSIONAL 300.
IF YOU HAVE  NOT  COMPLETED  AUTOMATIC  INSTALLATION,   YOU MUST
PERFORM A MANUAL BACKUP BEFORE USING VERIFY.  NOTE ALSO,  VERIFY
IS NOT SUPPORTED ON RX01 DISKETTES,    DECTAPE I OR II,   OR THE
PROFESSIONAL 325.

RT-11 SINGLE JOB MONITOR,  VOLUME ID  RT11A
BOOTED ON 23-SEP-86 AT 09:11:12 FROM FILE VM0:RT11SJ.SYS
@(EOF)
```

А теперь можно выключать диски — при условии, что вся необходимая информация находится на VM:. Если бы мы работали на ЭВМ «Электроника-79» или СМ-1420, у которых объем памяти достигает 4М байт, то можно было бы переписать на VM: почти полтора жестких диска RK:, выключить

их, чтобы не изнашивались, а поработав — скопировать сделанное назад, на RK:. Но если что-нибудь случится, например включат электричество,— вся работа может пропасть, и поэтому, прежде чем решаться на подобные эксперименты, нужно убедиться в надежности аппаратуры и электропитания в помещении.

Использование виртуальной памяти дает и другие удобства. Многие программы создают во время своей работы промежуточные файлы, которые не нужны после получения окончательных результатов, и эти промежуточные файлы разбивают на кусочки пространство на диске. Пусть занимаются этим где-нибудь на стороне, лучше всего — на устройстве VM:.

8.2. Логические диски

Мы не часто будем работать на «сотке» — пока нам более доступен ДВК, поэтому подготовим новую систему для него.

От старой она будет отличаться лишь дополнительными программами, которые мы скопировали с архивного диска. Мелкие улучшения, которые в ней произошли, будем отмечать по мере их использования. Так что все, что мы говорили до сих пор, верно и для новой системы.

Впрочем, у нее есть и еще одно отличие от старой — размер диска. Это в нашей власти — менять ее возможности и носители. Но так было не всегда. До появления V4.0 каждый монитор «знал» только свое устройство и такой простой возможности переходить от устройства к устройству не было. В особенности это касается тех версий ОС ДВК, которые имеют монитор MXMNSJ.SUS, соответствующий RT-11 V3.0.

Но к делу. Для начала создадим виртуальную модель нашего гибкого диска прямо на рабочем жестком диске. Он надежнее и будет хранить эталон нашей системы. Вспомним полный объем гибкого диска — 494. блока вместе с каталогом.

Наша задача формулируется просто — необходимо создать на диске RK3: файл размером 494. блока; пусть он для начала находится на любом свободном месте. Делается это такой командой:

```
.CREATE/ALLOCATE:494.
RK3:DVK2M.DSK⟨CR⟩
```

Файл будет называться так же, как и комплекс, на котором мы в основном работаем, а его тип говорит о том, что это будет диск. Нужный размер зарезервируется ключом /ALLOCATE.

Посмотрим, что получилось.

```
.DIR/FU/BL/VOL RK3:DVK*⟨CR⟩
14—MAR—88
VOLUME ID: RT-11SJ V5.1
OWNER  :
DVK2M  .DSK  494  14—MAR—88  425
UNUSED 3881 919
1 FILES, 494 BLOCKS
3881 FREE BLOCKS
```

В каталоге есть запись о файле — именно запись (здесь можно посоветовать вернуться к первой главе). В действительности файла как такового нет. Мы не знаем, как организована информация в той области диска, на которую указывает запись в каталоге. Пока еще нет адекватного соответствия между записью о файле и файлом как объектом.

Сейчас нужно как-то связать область на диске, используя, конечно, запись о ней в каталоге, с устройством LD*n:* (logical disk) с каким-нибудь номером. Как это сделать?

С устройством LD: можно обращаться как с диском. Поэтому присмотримся,

как устанавливаются большинство дисков и диск-пакетов.

А устанавливаются они сверху. И опять — за словарь. Сначала найдем слово «устанавливать». Есть много его вариантов, отражающих разные стороны жизни, но технических терминов только два — mount и install.

У слова install основное значение — «официальное введение в должность»; анализируя остальные значения, которых достаточно много, можно сказать, что обобщенный смысл слова install — устанавливать на место, вводя (водворяя, помещая) вовнутрь.

Это не то, что нам надо. Перекинем пару сотен страниц и найдем слово mount. У него значений еще больше, все они так или иначе связаны с горой, нужное нам — «устанавливать на возвышение». Можно в шутку сказать, что логический диск «взгромоздился» на жесткий диск.

Теперь осталось определить синтаксис команды. Файл, который должен быть связан с логическим диском, находится на жестком диске. Поэтому логический диск мы должны «взгромоздить» на файл и, соответственно, на жесткий диск. Скажем эту фразу полностью — «взгромоздить» LD4: на файл RK3:DVK2M.DSK.

.MOUNT LD4: RK3:DVK2M.DSK⟨CR⟩

и сразу посмотрим, что изменилось в каталоге.

.DIR/FU/BL/VOL BK3:DVK⟨CR⟩*
14—MAR—88
VOLUME ID: RT-11SJ V5.1
OWNER :
DVK2M .DSK 494P 14—MAR—88 425
UNUSED 3881 919
1 FILES, 494 BLOCKS
3881 FREE BLOCKS

Ничего не изменилось, кроме одного признака — файл стал защищенным, т. е. его нельзя удалить. Но система теперь знает, что файл RK3:DVK2M.DSK может рассматриваться как устройство LD4:, а значит, можно повторить все операции по инициализации файловой системы на этом устройстве.

Так и сделаем, а после скопируем все файлы нашей рабочей системы на устройство LD4:.

.INITIALIZE/VOLUMEID LD4:⟨CR⟩
LD4:/INITIALIZE — ARE YOU SURE? *Y ⟨CR⟩*
OWNER NAME: АРХАНГЕЛСКИЙ ⟨CR⟩
.COPY/SYSTEM/NOLOG RK3: LD4:⟨CR⟩
?PIP—F—DEVICE FULL LD4:DVK2M.DSK

Ого, система попыталась скопировать сам файл логического диска на устройство LD4:. Ведь мы опять дали групповую спецификацию файлов. Но ничего страшного не случилось: ведь файл DVK2M.DSK расположен последним на диске RK3:, и все, что нам нужно было скопировать, вошло.

Это тот случай, когда на фатальные ошибки можно смотреть спокойно. Кроме того, иногда для упрощения команды можно использовать такие приемы.

Теперь нужно записать загрузчик монитора в начальные блоки тома. Вопрос в том, какой нужен загрузчик, какой монитор в первую очередь он должен загружать и с какого устройства.

Давайте подумаем сначала над таким вопросом: можем ли мы загрузить систему с логического диска при включении ЭВМ и что для этого нужно? Последовательность действия должна быть следующей:

аппаратный загрузчик загружает систему с какого-нибудь физического диска, в нашем случае с RK3:;

система должна узнать, с каким номером логического диска какой файл

связан (например, из команды MOUNT);

после этого система должна узнать, с какого логического диска нужно загружаться, ведь на физическом диске могут находиться несколько файлов, связанных с логическими дисками.

И т. д. и т. п.

Но ведь мы хотели только создать эталонную копию системы для ДВК, как бы слепок с 8-дюймовой дискеты, а не загружаться с этой копии. Можем ли мы записать на этот том загрузчик того устройства, на котором система будет реально существовать?

Вот это уже другое дело, для этого достаточно указать в команде копирования загрузчика имя устройства (из двух символов). Такая возможность появилась только с версии 4.0.

.COPY/BOOT:DX:LD4:RT11SJ LD4:⟨CR⟩

И опять необходимо подчеркнуть — следует копировать загрузчик монитора, находящегося на том же устройстве, в начальные блоки которого он копируется.

Последнее, что нужно сделать,— скопировать устройство LD4: на устройство DX: и загрузится с устройства DX0:. Это уже несложно.

.COPY/DEVICE LD4: DX0:⟨CR⟩
DXO:/COPY — ARE YOU SURE? *Y⟨CR⟩*
.BOOT DXO:⟨CR⟩

Когда после красивого текста система поставила точку, можно сказать, что все сделано правильно и мы «пересели на другую табуретку», т. е. в качестве системного устройства теперь используется DX0:.

8.3. Обмани систему, или Форма без содержания

Для начала попробуем выполнить некорректное действие — скопируем устройство LD4: на устройство VM:.

.INI/NOQUERY VM:⟨CR⟩
.COPY/DEVICE LD4: VM:⟨CR⟩
OUTPUT DEVICE TOO SMALL — ARE YOU SURE? *Y⟨CR⟩*

...Скопировали!!! Но ведь все файлы не должны были поместиться, мы уже с этим сталкивались! Ну-ка, посмотрим каталог.

.DIR/FU/BL/VOL VM:⟨CR⟩

```
21-MAR-88
VOLUME ID: RT-11  V5.1B
OWNER    :
SWAP    .SYS   26P 13-JAN-84    8      RT11SJ.SYS   73P 13-JAN-84   34
TT      .SYS    2P 13-JAN-84  107      LD      .SYS    8P 13-JAN-84  109
DX      .SYS    4P 13-JAN-84  117      VM      .SYS    3P 13-JAN-84  121
RK      .SYS    3P 13-JAN-84  124      DP      .SYS    3P 13-JAN-84  127
MT      .SYS    9P 13-JAN-84  130      LP      .SYS    2P 13-JAN-84  139
PC      .SYS    2P 13-JAN-84  141      NL      .SYS    2P 13-JAN-84  143
SL      .SYS   13P 13-JAN-84  145      PIP     .SAV   29P 13-JAN-84  158
DUP     .SAV   45P 13-JAN-84  187      DIR     .SAV   19P 13-JAN-84  232
LET     .SAV    5P 13-JAN-84  251      K52     .SAV   55P 13-JAN-84  256
DUMP    .SAV    8P 13-JAN-84  311      EDIT    .SAV   19P 13-JAN-84  319
FORMAT.SAV   21P 13-JAN-84  338      PATCH   .SAV    9P 13-JAN-84  359
STARTS.COM    1P 13-JAN-84  368      VSNOTE.TXT    2P 13-JAN-84  369
IND     .SAV   51P 13-JAN-84  371      DATIME.COM    1P 13-JAN-84  422
( UNUSED )   71              423
 26 FILES, 415 BLOCKS
 71 FREE BLOCKS
```

Это ситуация, когда правая рука не ведает, что делает левая. Программа DUP скопировала большое устройство на маленькое — столько, сколько влезло, т. е. последних файлов, указанных в каталоге, не существует. Но программа DIR работает только с каталогами и показывает нам, нисколько над этим не задумываясь, каталог устройства, не соответствующий реальному.

Чтобы лучше понять это (а точнее, прочувствовать), давайте создадим в виде файла логического диска только каталог LD4:.

Для этого необходимо сконструировать команду. Мы опять копируем устройство на устройство, поэтому используем команду COPY/DEVICE. Но так как мы копируем не все устройство, а только часть его, то используем ключи /START и /END для входного устройства, а в качестве аргументов — номера блоков начала и конца копируемой области. При этом не забудем, что в результате мы должны получить файл, который потом можно будет связать с логическим диском. Для того чтобы в каталоге выходного устройства была создана запись с указанным нами именем и типом файла, после спецификации выходного файла укажем ключ /FILES.

.COPY/DEV LD4/START:0/END:12. RK3:KATAL.DSK/FILES⟨CR⟩
RK3:KATAL.DSK/COPY — ARE YOU SURE? *Y⟨CR⟩*
.DIR/FU/BL/VOL RK3:⟨CR⟩

```
21-MAR-88
VOLUME ID: RT-11SJ V5.1
OWNER    :
SWAP    .SYS   26P 13-JAN-84   10      RT11SJ.SYS   73P 13-JAN-84   36
TT      .SYS    2P 13-JAN-84  109      LD      .SYS    8P 13-JAN-84  111
DX      .SYS    4P 13-JAN-84  119      VM      .SYS    3P 13-JAN-84  123
RK      .SYS    3P 13-JAN-84  126      DP      .SYS    3P 13-JAN-84  129
MT      .SYS    9P 13-JAN-84  132      LP      .SYS    2P 13-JAN-84  141
PC      .SYS    2P 13-JAN-84  143      NL      .SYS    2P 13-JAN-84  145
SL      .SYS   13P 13-JAN-84  147      PIP     .SAV   29P 13-JAN-84  160
DUP     .SAV   45P 13-JAN-84  189      DIR     .SAV   19P 13-JAN-84  234
```

```
LET    .SAV     5P 13-JAN-84 253     K52    .SAV    55P 13-JAN-84 258
DUMP   .SAV     8P 13-JAN-84 313     EDIT   .SAV    19P 13-JAN-84 321
FORMAT.SAV    21P 13-JAN-84 340     PATCH  .SAV     9P 13-JAN-84 361
STARTS.COM     1P 13-JAN-84 370     V5NOTE.TXT     2P 13-JAN-84 371
IND    .SAV    51P 13-JAN-84 373     DATIME.COM     1P 13-JAN-84 424
DVK2M  .DSK   494P 16-MAR-88 425     KATAL  .DSK   13  21-MAR-88 919
( UNUSED ) 3868              932
 28 FILES, 922 BLOCKS
 3868 FREE BLOCKS
```

Теперь смонтируем получившийся файл (размером 13. блоков) с логическим диском, например LD2:, и посмотрим его каталог.

.MOUNT LD2: RK3:KATAL⟨CR⟩
.DIR/FAST/FULL LD2:⟨CR⟩

```
 21-MAR-88
SWAP   .SYS    RT11SJ.SYS    TT     .SYS    LD     .SYS    DX     .SYS
VM     .SYS    RK     .SYS   DP     .SYS    MT     .SYS    LP     .SYS
PC     .SYS    ML     .SYS   SL     .SYS    PIP    .SAV    DUP    .SAV
DIR    .SAV    LET    .SAV   K52    .SAV    DUMP   .SAV    EDIT   .SAV
FORMAT.SAV     PATCH  .SAV   STARTS.COM     V5NOTE.TXT     IND    .SAV
DATIME.COM     ( UNUSED )
 26 FILES, 415 BLOCKS
 71 FREE BLOCKS
```

Мы знаем, что файл имеет размер 13. блоков. Однако каталог показывает, что в нем имеются все (якобы) файлы и размер этого логического диска — 494. блока. Конечно, это игра. Зачем она нужна?

Полученные значения пригодятся, если придется заниматься восстановлением записанной на диске информации.

Если информация большого объема (например, на диск-пакете) хранится как бы упакованной в логические диски, то ее восстановление во многом упрощается. Начало логического диска найти очень просто — это те шесть плюс, как минимум, два блоков каталога, которые мы рассматривали в пятой главе. Найдя начало файла логического диска (пусть это будет блок 600.), можно в основном каталоге создать запись об этом файле, указав его размер, например 40. блоков,— это мы уже умеем — и связать этот файл с каким-либо номером логического диска. Все эти действия не причиняют вреда той информации, до которой мы добраться пока не можем. Посмотрев каталог, мы легко узнаем действительный размер нашего логического диска. А после этого можно дать «задний ход» — удалить нашу фиктивную запись в каталоге основного диска и создать новую, с действительным размером файла. Команду для этого придумать не трудно — достаточно добавить ключ /START:NBLK, указывающий абсолютный номер блока на томе:

.CREATE RK3:PRIMER.DSK/START:600./ALLOCATE:40.⟨CR⟩

Здесь ключи указаны не в команде, а после спецификации файла. Это не является требованием системы, а просто логичнее с точки зрения естественного языка.

А можно ли проделать то же за один прием? Ведь в основном каталоге это место считается свободным. Можем ли мы увеличить созданный нами файл до нужного размера?

Попробуем использовать ключ /EXTENSION, использовав в качестве параметра число, указывающее, на сколько блоков нужно увеличить размер файла:

.CREATE/EXTENSION:481. RK3:KATAL.DSK⟨CR⟩

Проверим:

.DIR/FU/BL/VOL RK3:KATAL ⟨CR⟩

```
21—MAR—88
VOLUME ID: RT—11SJ V5.1
OWNER    :
KATAL  .DSK  494P  21—MAR—88  919
⟨UNUSED⟩ 3387 1413
1 FILES, 494 BLOCKS
3387 FREE BLOCKS
```

Проделаем то же без помощи KMON.

Для создания записи о файле RK3:DVK2M.DSK можно бы воспользоваться программой DUP, но, так как аналога для ключа /ALLOCATE нет, сделаем так:

.RUN SY:DUP⟨CR⟩
**RK3:DVK2M.DSK[494.]=/C⟨CR⟩*

Число в квадратных скобках указывает на размер будущего файла.

Чтобы указать и номер блока, с которого будет начинаться файл, нужно добавить ключ /G:N

.RUN SY:DUP⟨CR⟩
**RK3:PRIMER.DSK[40]=/C/G:600.⟨CR⟩*

До версии 4.0 команды CREATE еще не было, и эта операция выполнялась иначе:

.RUN SY:DUP⟨CR⟩
**RK3:PRIMER.DSK=/C:600.:40.⟨CR⟩*

причем размер файла (в данном случае 40. блоков) по умолчанию принимался равным одному блоку и мог не указываться.

Ключ для расширения оставался неизменным во всех версиях:

.RUN SY:DUP⟨CR⟩
**RK3:KATAL.DSK=/T:481.⟨CR⟩*

Что же касается копирования в образе устройства, то здесь при появлении версии 4.0 также произошли изменения. Операцию, которую раньше выполняли в два этапа:

.RUN SY:DUP⟨CR⟩
**RK3:A=RK3:/I:425.:437.:919.⟨CR⟩*
RK3/COPY — ARE YOU SURE? *Y⟨CR⟩*
**RK3:KATALDSK/C/G:425.:12.⟨CR⟩*

теперь можно выполнить за один прием. Но сначала скажем, что при копировании для выходного устройства необходимо было указывать фиктивное имя файла A, а в качестве параметров для ключа /I (ignore — образ) нужно было последовательно указать начальный номер блока входного устройства, конечный номер блока входного устройства и начальный номер блока выходного устройства. В версиях от V4.0 и далее эта операция выполняется много проще:

.RUN SY:DUP⟨CR⟩
**RK3:KATALDSK[/G:919.]=LD4:/I/G:0/E:12./F⟨CR⟩*
RK3:/COPY — ARE YOU SURE? *Y⟨CR⟩*

здесь используется способность интерпретатора командной строки (CSI) (начиная с версии 4.0) различать, к входному или выходному файлу относится ключ. Это позволяет сделать команду логичнее и использовать меньший арсенал ключей. Ключ /F=FILES теперь указывает также на то, что копируется файл.

8.4. Копирование файлов на магнитную ленту

В журнальном варианте этот раздел опускается, так как речь в нем идет о накопителе на магнитной ленте, не использующемся совместно с ПЭВМ.

8.5. Типовой набор операций с файлом

При работе с файлами используется некоторый типовой набор операций. Вы уже заметили, что не все команды COPY обслуживает программа PIP. В каждом конкретном случае нужное решение принимает KMON. Когда же власть в свои руки берет пользователь, он должен знать, чем работать.

Если нужно изменить структуру каталога — добавить или переставить записи,

записать информацию в начальные блоки тома, то используется программа DUP.

Если нужно переместить файл или изменить некоторые поля в записи о файле (не перемещая ее в каталоге), то следует использовать программу PIP.

Отвлечемся от каталога и начальных блоков тома. Займемся только файлом. Что мы можем сделать с ним как с объектом, содержащим информацию? Пока только одно — сделать его копию, т. е. скопировать информацию из одного файла в другой (не будем пока говорить о преобразовании информации, потому что это особая большая тема). Мы даже не можем уничтожить информацию в прямом смысле слова. Мы можем только заменить одну информацию другой, может быть «нулевой», бессмысленной.

Итак, о копировании. В RT-11 (как и во многих других системах) имеются устройства, которые могут быть для удобства пользователя выделены в смысле возможностей копирования с одного устройства на другое.

Первое из таких устройств — системная консоль, или системный терминал. Конечно, мы можем дать для передачи файла на него обычную команду

COPY DX0:V5NOTE.TXT TT:

Однако можно поступить и иначе. В давние времена в качестве системного терминала подключался телетайп, а так как системный терминал находится близко от машины, то приставку tele можно убрать и дать команду

TYPE DX0:V5NOTE.TXT

которая по результату полностью соответствует предыдущей.

Практически в каждой системе используется печатающее устройство. Чем оно отличается от терминала? Если мы обратимся к русско-английскому словарю, то найдем, что для слова *печатать* есть два значения — type и print. С чем это связано?

Главное значение слова type — тип. В «печатать» оно превратилось с изобретением пишущей машинки — на ней писали «типовыми» буквами.

Слово print произошло от другого вида деятельности — печатания книг. Вычислительная техника для вывода информации требовала более быстродействующих печатающих устройств, чем пишущие машинки. Фирма IBM, пионер в данной области, начала использовать печатающие барабаны, состоящие из дисков с полным набором литер. Строка, по существу, печаталась за один удар. Эти устройства были названы принтерами. Вы уже догадались, что команду
COPY DX0:V5NOTE.TXT LP:
можно заменить на
PRINT DX0:V5NOTE.TXT

Таким образом, TYPE и PRINT применяются только для удобства пользователя и, в сущности, являются вариантами команды COPY. Однако они по умолчанию требуют тип файла .LST, чего не требует команда COPY. С помощью программы PIP эти команды выполняются одинаково:
.RUN SY:PIP⟨CR⟩
**LP:=DX0:V5NOTE.TXT⟨CR⟩*
или
**TT:=DX:V5NOTE.TXT⟨CR⟩*

Среди операций над записями каталога есть переименование файла:

RENAME DX0:SAMPLE.KED
DX0:EXEMIN.K52⟨CR⟩

Синтаксис этой команды нам привычен, и ясно, что сначала записываются старое имя и тип файла, а затем — новое имя. Это можно также выполнить с помощью программы PIP без посредников, используя ключ /R:
.RUN SY:PIP⟨CR⟩
**DX0:EXEMIN.K52=DX0:SAMPLE.KED/R⟨CR⟩*
Следует напомнить, что эта команда не применима к магнитным лентам и вообще к устройствам, не имеющим стандартного каталога.

В данном случае информация никуда не перемещалась — изменилось лишь содержание одного поля в записи о файле. Попробуем менять содержание других полей, а прежде вспомним их.
SIGN — признаки состояния файла
NAME.TYP — имя и тип файла
SIZE — размер файла
STATUS — состояние файла для системы
DATE — дата создания файла

Основное значение признака состояния, благодаря которому мы можем работать с файлом,— «постоянный». Если мы изменим значение на «пустой», то как бы удалим этот файл. Такую операцию проделывает команда DELETE.

После команды
DELETE VM:K52.SAV
которую мы применяли в предыдущей главе, мы видели след этого файла в распечатке каталога удаленных файлов, т. е. файл как бы вычеркнут, но так, что его можно увидеть. Основной ключ для выполнения этой операции программой PIP будет /D.
.RUN SY:PIP⟨CR⟩
**VM:K52.SAV/D⟨CR⟩*
В первых версиях системы нужно было при выполнении групповой операции явно указывать ключ /Q (query). Но в V5.0 и далее при удалении файлов система будет задавать вопрос во всех случаях, когда не указана явно полная спецификация файла. Иначе говоря, если задана групповая спецификация и подразумевается список файлов, подлежащих удалению, то нужно будет подтверждать выполнение операции для каждого из них. Такой подход кроме помощи забывчивым дает и дополнительные удобства: ключ /SYSTEM не нужен, если задана полная спецификация файла.

Изменить признак состояния файла «защищенный» на «незащищенный», или попросту снять защиту, мы уже пробовали командой UNPROTECT. А чтобы защитить файлы, достаточно дать команду
.PROTECT DX:.SAV⟨CR⟩

Признак состояния файла «защищенный» появился в версии 4.0, в которой

команд PROTECT и UNPROTECT еще не было, а так как изменение признака является переименованием, то делалось это следующим образом:

.RENAME/PROTECT DX:.SAV⟨CR⟩

С помощью программы PIP эти команды выполняются одинаковым образом:

.RUN SY:PIP⟨CR⟩
**DX:.SAV=DX0:.SAV/R/F⟨CR⟩*

для установки защиты и

.RUN SY:PIP⟨CR⟩
**DX:.SAV=DX0:.SAV/R/Z⟨CR⟩*

для снятия защиты.

Этот типовой набор команд имеет, естественно, и общий набор ключей. Их наличие или отсутствие в конкретной команде диктуется логикой ее выполнения. Команда COPY имеет самый большой набор ключей, который включает почти все ключи, используемые в типовом наборе команд. Поэтому начнем с команды COPY.

8.6. Выбор исходных данных

Вспомним, как работали с программой DIR. Для облегчения работы мы распечатывали каталог с указанной датой, до или после указанной даты. Можно просто выбрать новые файлы. Мы уже умеем создавать файлы (или хотя бы записи о них), а система считает новыми файлы, у которых дата создания совпадает с текущей датой системы.

А раз наши потребности при работе с программами PIP и DIR совпадают, то вероятно, что этот набор ключей для них общий:

BEFORE [:DD:MMM:YY] — до указанной даты;
DATE [:DD:MMM:YY] — с указанной датой;
NEWFILES — с текущей системной датой;
SINCE [:DD:MMM:YY] — после указанной даты.

Копируя файл, мы можем воздействовать на запись в каталоге о нем. И если мы можем выбирать файлы по указанной дате, то, наверное, можно при копировании установить новую дату для выходного файла. Нужным для этого ключом будет /SETDATE[:DD:MMM:YY]. Если же дату опустить, то система установит текущую системную дату.

Эти ключи действительны для всех перечисленных команд, однако понятно, что ключ /SETDAT не может быть применен с командами DELETE, PRINT и TYPE. Команда DELETE вычеркивает запись из каталога, поэтому ничего не может установить. Команды PRINT и TYPE копируют файлы на устройства, не имеющие каталогов.

Укажем соответствующие ключи CSI, которые нужно применять с программой PIP:

/BEFORE [:DD:MMM:YY] — /J:DD:MMM:YY
/DATE [:DD:MMM:YY] — /C:DD:MMM:YYY
/NEWFILES — /C
/SINCE [: DD:MMM:YY] — /I:DD:MMM:YY
/SETDAT [:DD:MMM:YY] — /T:DD:MMM:YY

8.7. Какие файлы мы копируем

Не касаясь информации в файлах, попытаемся сделать что-нибудь посложнее простого копирования. Например, возьмем два (или несколько) входных файла и скопируем их в один выходной. Простейший способ сделать это

COPY DX0:(FILE1+FILE2).TXT
DX1:BIGEIL.TWO

появился только начиная с версии 4.0. А до этого? Если мы вспомним язык математики, то там операция соединения (или объединения) называется конкатенацией. Отсюда другая форма этой команды:

COPY/CONCATENATE DX0:FILE1.TXT, DX0:FILE2.TXT DX1:BIGEIL.TWO

Обычно ключ /CONCATENATE используется с групповой спецификацией, когда нужно большую группу файлов (как правило, имеющих общий признак) «слить» в один файл, например, так:

COPY/CONCATENATE LD0:IGL%%.DOC MT0:BOOCK.DOC

при этом следует учесть, что файлы будут копироваться в порядке, указанном в каталоге, а также то, что выходной файл не может иметь групповую спецификацию.

Таким путем мы соединяем копируемые файлы, не учитывая их наполнения: отображаем один входной файл на выходной, а затем впритык к нему (по

границе блоков) отображаем второй входной файл. Это действует (по умолчанию) ключ /IMAGE — образ. А как быть с текстовыми файлами? В третьей главе мы получали два маленьких текстовых файла — DX0FIL.DIR и DXFILE.DIR. Если их объединить таким способом, то между их текстовыми частями (в конце второго блока) получится вставка из нулевых кодов.

.COPY DX:(DX0FIL+DXFILE).DIR DX0:TWOFIL.TXT⟨CR⟩
.DUMP/TERMINAL DX:TWOFIL.TXT⟨CR⟩

```
DX0:TWOFIL.TXT
BLOCK NUMBER  00000
000/ 006440 . . . 027040 054523 020123 020040 * ..SWAP  .SYS    *
020/ 031040 . . . 060515 026562 031470 020040 * 26  10-MAR-83   *
  .
  .
  .
740/ 020040 . . . 030462 046455 074541 034055 *     9P 21-MAY-8*
760/ 006465 . . . 042523 020104 020076 020040 *5..( UNUSED )   *

BLOCK NUMBER  00001
000/ 030461 . . . 020040 020040 020040 020040 *115              *
020/ 020040 . . . 051525 042105 037040 020040 *    ( UNUSED )   *
  .
  .
  .
740/ 000000 . . . 000000 000000 000000 000000 *.................*
760/ 000000 . . . 000000 000000 000000 000000 *.................*

BLOCK NUMBER  00002
000/ 006440 . . . 027040 054523 020123 020040 * ..SWAP  .SYS    *
020/ 031040 . . . 060515 026562 031470 020040 * 26  10-MAR-83   *
  .
  .
  .
740/ 020040 . . . 030462 046455 074541 034055 *     9P 21-MAY-8*
760/ 006465 . . . 042523 020104 020076 020040 *5..( UNUSED )   *

BLOCK NUMBER  00003
000/ 030461 . . . 020040 020040 020040 020040 *115              *
020/ 020040 . . . 051525 042105 037040 020040 *    ( UNUSED )   *
040/ 030440 . . . 020040 020040 020040 006440 * 115            .*
060/ 020012 . . . 071545 020054 032462 020066 *. 16 FILES, 256 *
100/ 066102 . . . 031040 030063 043040 062562 *BLOCKS.. 230 FRE*
120/ 020145 . . . 005015 000000 000000 000000 *E BLOCKS.........*
140/ 000000 . . . 000000 000000 000000 000000 *.................*
160/ 000000 . . . 000000 000000 000000 000000 *.................*
200/ 000000 . . . 000000 000000 000000 000000 *.................*
220/ 000000 . . . 000000 000000 000000 000000 *.................*
240/ 000000 . . . 000000 000000 000000 000000 *.................*
260/ 000000 . . . 000000 000000 000000 000000 *.................*
300/ 000000 . . . 000000 000000 000000 000000 *.................*
320/ 000000 . . . 000000 000000 000000 000000 *.................*
340/ 000000 . . . 000000 000000 000000 000000 *.................*
360/ 000000 . . . 000000 000000 000000 000000 *.................*
400/ 000000 . . . 000000 000000 000000 000000 *.................*
```

Чтобы этой дырки не было, попробуем сказать системе, что мы копируем файлы в коде ASCII и поэтому коды с нулевым значением нужно при чтении игнорировать.

.COPY/ASCII DX1:(DX0FIL+DXFILE).DIR DX1:TWOFIL.TXT⟨CR⟩
.DUMP/TERMINAL DX:TWOFIL.TXT⟨CR⟩

```
BLOCK NUMBER  00001
000/ 030461 . . . 020040 020040 020040 020040 *115              *
020/ 020040 . . . 051525 042105 037040 020040 *    ( UNUSED )   *
040/ 030440 . . . 020040 020040 020040 006440 * 115            .*
060/ 020012 . . . 071545 020054 032462 020066 *. 16 FILES, 256 *
100/ 066102 . . . 031040 030063 043040 062562 *BLOCKS.. 230 FRE*
120/ 020145 . . . 005015 006440 051412 040527 *E BLOCKS..  ..SWA*
140/ 020120 . . . 020040 031040 020066 030440 *P  .SYS    26  1*
160/ 026460 . . . 020040 020040 020040 052122 *0-MAR-83       RT*
200/ 030461 . . . 020040 020040 031467 020040 *11SJ.SYS    73  *
220/ 030461 . . . 006464 052012 020124 020040 *11-MAR-84..TT   *
240/ 027040 . . . 020040 020062 030440 026460 * .SYS     2  10-*
260/ 060515 . . . 020040 020040 054104 020040 *MAR-83       DX  *
300/ 020040 . . . 020040 031440 020040 033060 *  .SYS     3  06*
320/ 043055 . . . 051412 020114 020040 027040 *-FEB-84..SL     .*
340/ 054523 . . . 020063 030040 026466 062506 *SYS    13  06-FE*
360/ 026542 . . . 020040 050114 020040 020040 *B-84       LP    *
400/ 051456 . . . 031040 020040 030463 046455 *.SYS     2  31-M*
420/ 071141 . . . 020130 020040 027040 054523 *AR-84..MX    .SY*
440/ 020123 . . . 031040 026471 061517 026564 *S    10  29-OCT-*
460/ 032070 . . . 046116 020040 020040 051456 *84       NL    .S*
500/ 051531 . . . 020040 032061 047055 073157 *YS      2  14-NOV*
520/ 034055 . . . 020040 027040 054523 020123 *-85..RK    .SYS *
540/ 020040 . . . 026464 067516 026566 032470 *     3  14-NOV-85*
560/ 020040 . . . 020040 020040 051456 051531 *       MT    .SYS*
600/ 020040 . . . 032061 047055 073157 034055 *     9  14-NOV-8*
620/ 006465 . . . 027040 054523 020123 020040 *5..DP    .SYS    *
640/ 020040 . . . 067516 026566 032470 020040 *  3  14-NOV-85  *
660/ 020040 . . . 020040 051456 051531 020040 *    LD    .SYS  *
700/ 020040 . . . 043055 061145 034055 006464 *   BP 06-FEB-84.*
720/ 050012 . . . 040523 020126 020040 031040 *.PIP    .SAV    2*
740/ 020071 . . . 026562 031470 020040 020040 *9  10-MAR-83     *
760/ 020040 . . . 051456 053101 020040 020040 *  DUP    .SAV    *

BLOCK NUMBER  00002
000/ 032464 . . . 071141 034055 006463 042012 *45  10-MAR-83..D*
020/ 051111 . . . 020126 020040 030440 020071 *IR    .SAV    19 *
040/ 030440 . . . 031470 020040 020040 020040 * 10-MAR-83       *
060/ 052504 . . . 053101 020040 020040 034440 *DUMP   .SAV     9*
100/ 020120 . . . 034055 006465 036012 052440 *P 21-MAY-85..( U*
120/ 052516 . . . 020040 030461 020065 020040 *NUSED )   115   *
140/ 020040 . . . 020040 020040 020040 020074 *               ( *
160/ 047125 . . . 020040 030440 032461 020040 *UNUSED )   115   *
200/ 020040 . . . 006440 020012 033061 043040 *          .. 16 F*
220/ 066151 . . . 020066 066102 061557 071553 *ILES, 256 BLOCKS*
240/ 005015 . . . 062562 020145 066142 061557 *.. 230 FREE BLOC*
260/ 071553 . . . 000000 000000 000000 000000 *KS.............*
300/ 000000 . . . 000000 000000 000000 000000 *.................*
320/ 000000 . . . 000000 000000 000000 000000 *.................*
340/ 000000 . . . 000000 000000 000000 000000 *.................*
360/ 000000 . . . 000000 000000 000000 000000 *.................*
400/ 000000 . . . 000000 000000 000000 000000 *.................*
420/ 000000 . . . 000000 000000 000000 000000 *.................*
440/ 000000 . . . 000000 000000 000000 000000 *.................*
460/ 000000 . . . 000000 000000 000000 000000 *.................*
500/ 000000 . . . 000000 000000 000000 000000 *.................*
520/ 000000 . . . 000000 000000 000000 000000 *.................*
540/ 000000 . . . 000000 000000 000000 000000 *.................*
560/ 000000 . . . 000000 000000 000000 000000 *.................*
600/ 000000 . . . 000000 000000 000000 000000 *.................*
620/ 000000 . . . 000000 000000 000000 000000 *.................*
640/ 000000 . . . 000000 000000 000000 000000 *.................*
660/ 000000 . . . 000000 000000 000000 000000 *.................*
700/ 000000 . . . 000000 000000 000000 000000 *.................*
```

Приведена только часть реального дампа.

Итак, теперь это чистый текстовой файл.

При работе с текстовыми файлами нужно быть осторожным и помнить о свойствах кода ASCII. Поскольку символ в нем кодируется семью битами, программа PIP будет обнулять (для надежности) восьмой бит. Тексты, записанные на RADIX-50 или КОИ-8, это изуродует до неузнаваемости.

Нужно помнить и о том, что символ CTRL+Z принят в системе в качестве признака конца текста, и если он случайно встретится в середине входного файла, то будет воспринят как конец файла.

Своя особенность есть у файлов, записанных в специальном двоичном формате фирмы DEC (как правило, они имеют тип .OBJ или .LDA). Их

кусочки различной длины имеют в своем составе специальные контрольные коды, позволяющие проверить правильность соответствующего куска файла. Система поддерживает и это исключительное свойство, для чего нужно указать ключ /BINARY. В случае неверного контрольного кода программа PIP выдаст об этом сообщение.

Понятно, что этот набор ключей присущ только команде COPY. Команды PRINT и TYPE сами знают, что работают с кодом ASCII, а остальные при всем желании не могут использовать эту информацию.

Соответствующие однобуквенные ключи CSI для программы PIP:

/CONCATENATE — /U
/ASCII — /A
/IMAGE — нет
/BINARY — /B

8.8. Доверяй, но проверяй

Конечно, система как-то следит за правильностью выполнения операций копирования, но не вредно и самим присмотреть за этим. Нам уже знакомо слово VERIFY; оно может служить ключом и для программы PIP (начиная с версии 5.0).

При копировании PIP считывает с входного носителя в свой буфер некоторую порцию информации, а затем записывает ее из буфера на выходной носитель. Если задан ключ /VERIFY, то дополнительно происходит считывание с выходного носителя только что записанной порции и сравнение ее с образцом, находящимся в буфере. Оптимальные устройства для копирования с этим ключом — записывающие информацию блоками или его частями, такие, как RK:, DX:, DP:.

Проверка — это хорошо, но вот текст на русском языке, например, содержит избыточную информацию. Если в текстовом файле возникла ошибка, то, проигнорировав ее, мы спасем большую часть, а может быть, и весь текст. Ключ для этого — /IGNORE. Помните, что он не уживается с ключом /VERIFY, применять их вместе нельзя.

Если задан ключ /IGNORE, то встретившаяся во входном файле ошибка не вызывает прекращения копирования. Вместо этого появится сообщение об ошибке

?PIP—W—INPUT FILE ERROR

и копирование будет продолжаться. Более того, если на одном месте файла будет выдано только одно такое сообщение, то это означает, что системе удалось-таки прочитать этот блок.

Программа PIP копирует файлы большими кусками, как правило, по три блока и больше. Если возникнет ошибка, то не будет скопирован весь большой кусок. Поэтому хотелось бы копировать файл по одному блоку, несмотря на то что это будет заметно медленнее. Такой режим включается ключом /SLOWLY — медленно.

Сочетание ключей /SLOWLY/VERIFY очень эффективно при копировании на накопителях с поблочной организацией, таких, как DX:, RK:, DP:, а сочетание /SLOWLY/VERIFY позволяет терять минимум информации при копировании с ошибками и особенно удобно для магнитных лент.

Однобуквенные аналоги этих ключей в программе PIP:

/VERIFY — /H
/IGNORE — /G
/SLOWLY — /S

8.9. Шило на мыло

Копирование файлов позволяет нам перенести информацию с одного диска на другой, но при этом на входном

томе останутся исходные файлы. А если мы копировали, чтобы собрать какую-то группу файлов вместе и при этом освободить место на входном томе? Тогда нам придется удалять скопированные файлы с входного тома. То, что это неудобно,— полбеды. Главное, что при большом количестве файлов можно допустить ошибку и удалить не то, что копировали. Хорошо бы совместить копирование и удаление. А делается это так:

```
.COPY/DELETE  DX0:DX0FIL  DX1:⟨CR⟩
      FILES COPIED/DELETED
DX0:DX0FIL.DIR TO DX1:DX0FIL.DIR
```

Мы уже привыкли, что в RT-11 все продумано и логично; не удивительно, что наши пожелания так легко выполняются. Кстати, удаление файла происходит только в случае успешного копирования на выходной том.

Если на выходном томе уже имеется файл с таким же именем и типом, то система по умолчанию использует знакомый нам ключ /REPLACE — заменить, т. е. после успешного копирования файл с тем же названием, который первоначально был на томе, удаляется, а скопированный получает признак состояния «постоянный».

А если тот файл был нам нужен? Ведь могли мы, копируя, забыть, что имя уже «занято»? Чтобы уберечься от потери, следует отметить умолчание, сказав COPY/NOREPLACE.

Все это работает, если на выходном томе достаточно места, чтобы разместить очередной файл. Так бывает не всегда, порой необходимо предварительно удалить файл с таким же именем и типом, чтобы на его место скопировать новый. Для этого используется команда COPY/PREREPLACE, особенно полезная при работе на малых конфигурациях систем, таких, как ДВК.

С командами PRINT и TAPE также можно использовать ключ /DELETE, что позволяет не задумываться над тем, какие страницы или главы книги уже распечатаны, а какие нет.

При переименовании файлов может оказаться, что файл с указанным именем и типом уже существует. Поэтому ключи /REPLACE и /NOREPLACE применимы и с командой RENAME.

Соответствующие однобуквенные ключи CSI для программы PIP:

```
/DELETE     — /D
/REPLACE    — нет
/NOREPLACE  — /N
/PREDELETE  — /O
```

8.10. Печать многих копий

Несмотря на то что команды PRINT и TYPE являются аналогами команды COPY, они имеют особенность, связанную с размножением текста на принтере. Если нам нужно получить много бумажных копий одного и того же документа, об этом нужно просто сказать системе с помощью /COPIES:NN, в качестве аргумента указав количество копий (десятичное, от 1 до 32), например:

```
.PRINT/COPIES:3  SY:V5NOTE.TXT⟨CR⟩
```

Мы получим на бумаге три красивых текста, которые видели при загрузке системы. Этот полезный ключ программа PIP понимает в его однобуквенном эквиваленте /K:NN.

8.11. Последние штрихи

Заканчивая тему копирования файлов, отметим, что программа PIP позволяет копировать и на однодисковых системах, хотя это и утомительно при больших объемах или числе файлов. Ключ /WAIT в этом случае превращается в /E.

Приведем возможные сокращения команд и имена программ, которые их выполняют.

COPY	COP	PIP
TYPE	TY	PIP
DELETE	DEL	PIP
PROTECT	PRO	PIP
UNPROTECT	UNPR	PIP
PRINT	PRI	PIP
RENAME	REN	PIP
BOOT	BO	DUP
COPY/BOOT	COP/BO	DUP
COPY/DEVICE	COP/DEV	DUP
SQUEEZE	SQ	DUP
INITIALIZE	INI	DUP

А вот полный список рассмотренных ключей.

Команда KMON	Сокращение для KMON	PIP	DUP
LOG	L	/W	—
QUERY	Q	/Q	—
REPLACE	REP	—	/R
PROTECTION	PRO	/F	—
CONCATENATE	C	/U	—
PREDELETE	PRE	/O	—
DELETED	DEL	/D	—
IGNORE	IG	/G	—
ASCII	AS	/A	—
BINARY	BI	/B	—
IMAGE	IM	—	—
POSITION	PO	/M:N	—
EXCLUDE	EX	/P	—
SLOWLY	SL	/S	—
NEWFILES	NE	/C	—
BEFORE	BEF	—	—
DATE	DA	—	—
SINCE	SI	—	—
SETDATE	SET	/T	—
SYSTEM	SY	/V	—
BOOT	BO	—	/U:XX
DEVICE	DEV	—	/I
VERIFY	VE	—	—
FILES	FI	—	/F
WAIT	W	/E	/W
START	ST	—	/G:N
ALLOCATE	ALL	—	—
END	EN	—	/E:N
OUTPUT	OU	—	—
BADBLOCKS	BAD	—	/K
VOLUMEID	VO	—	/V
BEGIN	BEG	—	—
RESTORE	RES	—	/D
FOREIGN	FO	—	/Q

Продолжение следует

В. МОНАХОВ

Фокал БК-0010 и его расширения

В нашем журнале (Информатика и образование. 1987. № 3) уже публиковались сведения о Фокале. К сожалению, описанная там версия отличается от Фокала БК-0010, интерпретатором которого оснащено подавляющее большинство выпущенных ПЭВМ «Электроника БК-0010», «Электроника БК-0010Ш» и «Электроника БК-0010.01» (далее — БК). В прилагаемой к БК документации также опущен ряд важных особенностей этого языка программирования. Поэтому возникла идея изложить современное состояние Фокала БК, включая все его известные расширения.

За последнее время Фокал БК (далее просто Фокал) не раз подвергался критике, а Минпрос СССР вообще отказался от использования его в школах, приняв за базовые языки Рапиру и Бейсик. Однако причиной критических выступлений зачастую была неосведомленность их авторов, ругавших некий примитивный язык. Таков ли Фокал на самом деле? Что представляет он собой сегодня вместе с расширениями?

Начнем с того, что Фокал принадлежит к операторным языкам высокого уровня, но, в отличие от других языков этого класса (таких, как Фортран или Бейсик), имеет очень простой синтаксис, быстро и легко осваивается даже непрофессиональными пользователями, способен к неограниченному расширению.

Он включает все операторы, присущие языку программирования высокого уровня: присваивания, условного и безусловного перехода, повторения (цикла), обращения к подпрограмме, ввода информации с клавиатуры и с магнитной ленты, вывода информации на экран дисплея и на магнитную ленту, а также обычный набор встроенных арифметических функций, включая любые функции, программируемые пользователем.

Среди возможностей Фокала:

функция прямого обращения к памяти ПЭВМ, выполняющая три операции (запись в память, чтение и конъюнкция содержимого памяти и маски);

развитые графические функции, позволяющие создавать на экране изображения с высоким разрешением (графический экран БК имеет поле 512×256 точек);

функции обработки символьной информации;

функция, обеспечивающая работу с параллельным программируемым портом ввода-вывода, к которому можно подключать различные внешние приборы, управляемые ПЭВМ;

развитые отладочные средства, включающие автоматическое обнаружение ошибки в программе и указание ее местонахождения и характера, трассировку программы при исполнении, операторы распечатки на экране дисплея текста программы и текущих значений всех переменных, оператор редактирования текста программы, оператор стирания части или всего текста программы, а также значений переменных, оператор вычисления размера свободной памяти, отведенной пользователю, оператор, выдающий справочную информацию о языке.

Расширения Фокала позволяют:

вводить с магнитной ленты или другого компьютера программы не только целиком, но и отдельными блоками, присоединяемыми к уже существующей в памяти программе;

перенумеровывать строки программы;

организовывать массивы плотно упакованных символьных переменных, целых и вещественных чисел и вводить функции для разнообразной обработки этих массивов;

использовать в Фокале подпрограммы, написанные в машинных командах.

Возможность использования машинных подпрограмм поддерживается целой серией системных программ для БК; в их числе несколько отладчиков, ассемблеров и реассемблеров.

Таким образом, по своим возможностям БК с Фокалом практически не уступает БК с Бейсиком, не говоря уже о ПЭВМ с Рапирой. В то же время он выгодно отличается от них компактностью и простотой, а главное, предоставлением пользователю большей памяти для размещения программы и данных. Дело в том, что Бейсик БК очень нерационально расходует память. Для работы программы он создает в ОЗУ некое подобие объектного кода, которое занимает почти столько же места, сколько и исходный текст программы, который также хранится. Таким образом, Бейсик БК почти вдвое уменьшает реально доступную пользователю оперативную память. Фактически на нем можно писать программы, которые вместе с переменными занимают объем не более 7К байтов в режиме полного экрана и 14К байтов — в режиме расширенной памяти. В БК с Фокалом даже при использовании самой крупной программы расширения (СФМ) в распоряжении пользователя остается соответственно 13 и 25К байтов. Если же учесть, что и сами программы на Фокале получаются почти в 1,5—2 раза короче, чем на Бейсике, то реальная экономия памяти окажется еще более ощутимой.

Нельзя сбрасывать со счетов и то, что для БК с Фокалом уже разработано полтысячи программ, позволяющих решать многие практические задачи школьной информатики, и накоплен значительный опыт программирования, которым неразумно было бы пренебречь.

Объем статьи не позволяет рассмотреть в деталях весь Фокал. Остановимся лишь на некоторых его особенностях, которые повышают эффективность программ или не нашли достаточного освещения в руководстве пользователя, прилагаемом к компьютеру.

О номерах строк. В руководстве указано, что номера строк должны быть нецелыми числами, лежащими в интервале от 1.01 до 99.99 с интервалом 0.01 и от 100.1 до 127.9 — с интервалом 0.1. Это не совсем точно. Так выводятся номера строк оператором WRITE, но в БК-0010 они представлены иначе. Каждый номер представляет собой целое число, занимающее в памяти два соседних байта и лежащее в интервале от 257 до 32 767. При выводе на экран оно делится на 256 и округляется до 3 или 4 значащих цифр. Таким образом, в действительности номер строки может изменяться в пределах от 1.00391 до 127.996 с интервалом 1/256. Однако пользоваться дополнительными номерами строк не рекомендуется: ведь при выводе на экран номера, например, строк 1.004, 1.008 и 1.012 будут округлены одинаково до 1.01, а это может затруднить отладку программы.

Имена переменных. Имена перемен-

ных могут состоять не только из заглавных, но и из строчных латинских букв. Это расширяет возможности применения однобуквенных имен, сокращающих текст программ, и позволяет применять имена, начинающиеся с буквы f.

Переменные с индексами в Фокале легко перепутать. Чтобы этого не случалось, при использовании в одной программе переменных с одинаковыми именами, но разным числом индексов нужно помнить, что в Фокале фактически существует только один тип переменных, а именно переменные с двумя индексами. Так, имена A, A(0) и A(0,0) характеризуют одну и ту же переменную. Вот примеры других совпадений:

A(130)≡(—126,0)

B(256)≡B(0,+1)

A(255)≡A(—1,0)

Это обусловлено тем, что в области памяти, отводимой Фокалом под переменную, индексы располагаются в двух соседних байтах. Для одноиндексной переменной под индекс используются оба байта, а для двухиндексных — младший байт под первый индекс, а старший — под второй. В связи с изложенным во избежание путаницы целесообразно для различных индексированных переменных использовать разные имена.

Арифметические выражения. Существенным отличием арифметики Фокала от обычных правил является то, что умножение здесь выполняется раньше, чем деление, т. е. в операторе SET A=B/C*D вначале будет вычислено C*D, а лишь затем B будет поделено на это число.

Знак ¬ возводит число, стоящее перед ним, в целую положительную степень. Если число, стоящее за знаком ¬, смешанное, то будет отброшена его дробная часть, а оставшаяся целая будет использована в качестве показателя степени, так что 2¬2.99999=4.

Ситуация, показанная в этом примере, является типичной для Фокала и нередко приводит к ошибкам. Если показатель степени вычисляется в результате каких-либо арифметических действий, то во избежание получения ошибочного результата необходимо к вычисленному показателю степени прибавлять небольшое число, например 0.1.

В арифметических выражениях полезно чаще пользоваться скобками, чем это обычно принято в алгебре; как говорят бывалые программисты, скобки всегда обходятся дешевле ошибок!

Встроенные функции. Аргументы всех встроенных функций, кроме FX и FP, могут быть числами, переменными или любыми арифметическими выражениями. Обычная точность вычисления функций составляет 5—6 верных значащих цифр. Исключением являются тригонометрические функции с большими аргументами, которые вычисляются практически неверно. Если возникает потребность в таких вычислениях, следует разрабатывать специальные программы.

Функция FITP(X) только для положительных чисел действительно выделяет целую часть. Для отрицательных чисел она выделяет целую часть абсолютного значения аргумента и затем добавляет к ней знак «минус», например FITR(—4.82)=—4.

Функция FRAN( ) является генератором псевдослучайных чисел, лежащих в интервале от —1 до +1. Утверждение о равномерности распределения генерируемых чисел в этом интервале не соответствует действительности. Для улучшения этой функции можно воспользоваться советом, описанным в [1].

Функция FCHR(A) предназначена для обработки символьной информации и программного управления БК. Наиболее часто эта функция применяется в следующих случаях:

вывод десятичных кодов символов на экран — для этого можно воспользоваться простой программой

```
1.1 T FCHR(FCHR(—1)),!; G
```

приостановка выполнения программы — для этого в нужное место программы вставляется оператор

```
X FCHR(—1)
```

который остановит выполнение программы до тех пор, пока не будет нажата какая-либо клавиша;

ввод кода символа с клавиатуры в память; здесь обычны две конструкции — с выводом изображения символа на экран

```
S Z=FCHR(FCHR(—1))
```

и без вывода
S Z=FCHR(—1)
в обоих случаях переменной Z будет присвоен код введенного с клавиатуры символа;

вывод символа из памяти на экран — для этой цели используется оператор вида
X FCHR(N)
где N — десятичный код выводимого символа; в принципе N может быть, как сказано выше, любым арифметическим выражением. Перед исполнением дробная часть его отбрасывается, а целая — используется в качестве кода символа;

выполнение управляющих команд — это обеспечивается оператором
X FCHR(N)
где N — код управляющей команды (см. табл. 1).

Функции FK, FT и FV могут нормально работать с любыми значениями аргументов. Смешанные значения координат перед использованием обрабатываются функцией FITR, а если их значения в функции FK выходят за пределы 63 и 23 соответственно, то они вычисляются по модулю 64 и 24 соответственно и используются в качестве аргументов. У функций FT и FV этого не происходит. Просто изображение получается видимым на экране только при абсциссе, лежащей от 0 до 511, и ординате — от 0 до 239. Если эти пределы будут нарушены, изображение или его часть выйдет за пределы экрана.

Еще одна особенность: последние координаты X и Y, использованные функциями FT или FV, сохраняются в памяти и не могут быть стерты оператором ERASE. Их можно стереть только перезапуском системы.

Функция FX позволяет обращаться к любой ячейке адресного пространства БК. Она имеет вид
FX (КОП, A, D)
где КОП — код операции:

—1 — происходит запись целого десятичного числа или значения переменной D непосредственно в память БК по адресу A;

0 — происходит логическое побитовое умножение (конъюнкция) содержимого ячейки с адресом A и целого числа D и присвоение функции FX результата этой операции;

1 — происходит чтение, т. е. присвоение функции FX значения содержимого ячейки с адресом A;

A — адрес ячейки памяти БК, в качестве которого может быть либо восьмеричное число от 0 до 177776 либо

Таблица 1

Управляющие коды БК-0010
(в скобках указано действие при повторном вводе кода)

Десятичный код	Клавиша	Выполняемая команда
7		Подается короткий звуковой сигнал
8	←	Курсор перемещается на одну позицию влево
12	СБР	Очистка экрана
13	ВВОД	Ввод набранной на экране информации в БК
19	ВС	Перемещает текст ниже курсора на одну строку вверх
20	ГТ	Перемещает текст от строки курсора на одну строку вниз
25	→	Курсор перемещается на одну позицию вправо
26	↑	То же — вверх
27	↓	То же — вниз
140	НР/РП	Переводит БК в режим расширенной памяти
145		Увеличивает яркость выводимых символов*
146		Уменьшает яркость выводимых символов*
148		Делает выводимые символы невидимыми, т. е. одинаковыми с фоном*
149	ГРАФ	Переводит БК в режим текстовой графики (или выводит из него)
150	ЗАП	Включает режим записи изображения
151	СТИР	Включает режим стирания изображения
154	КУРСОР	Гасит (или восстанавливает) курсор
155	32/64	Переводит экран из режима «64 символа на строке» в режим «32 символа в строке» (или обратно)
156	ИНВ. С	Инверсия выводимых символов
157	ИНВ. Э	Инверсия экрана

* Эти команды исполняются в режиме «32 символа в строке»; в таблице описано исполнение команд в черно-белом варианте БК.

десятичная переменная Фокала, принимающая значение от 0 до 65 534; адрес A не может быть арифметическим выражением;

D — десятичное число, переменная или арифметическое выражение, принимающее целочисленное значение от —32 768 до +32 767.

Функция FP предназначена для управления внешними устройствами, подключенными к параллельному программируемому порту ввода-вывода. Такими устройствами могут быть электрофицированные учебно-наглядные пособия, лабораторные установки и т. п. Эта функция имеет вид

FP(КОП, M)

где M — маска (восьмеричное число от 0 до 177 777 или переменная, принимающая целое десятичное значение в пределах от —32 768 до 32 767).

КОП — код операции:

0 — происходит чтение по маске содержимого регистра порта ввода, т. е. логическое поразрядное умножение (конъюнкция) содержимого регистра и маски и присвоение функции FP результата этой операции;

1 — происходит присвоение нулей тем двоичным разрядам числа, записанного в порте вывода, которые соответствуют разрядам маски с двоичными единицами;

2 — происходит присвоение единиц тем двоичным разрядам числа, записанного в порте вывода, которые соответствуют одноименным разрядам маски с двоичными единицами;

3 — происходит чтение по маске содержимого регистра порта вывода.

Операторы. Фокал имеет 19 операторов. Это ASK, COMMENT, DO, ERASE, FOR, GOTO, HELP, IF, KILL, LIBRARY, MODIFY, PASS, QUIT, RETURN, SET, TYPE, VACANT, WRITE, XECUTE. В отличие от других языков программирования в Фокале можно сокращать имя оператора до одной начальной буквы. Это делает программы более короткими и исполняемыми быстрее.

Имена операторов набирают только заглавными латинскими буквами и отделяют от номера строки и от последующих аргументов не менее чем одним пробелом.

В операторе ASK ввода информации с клавиатуры можно применять те же операнды, что и в операторе TYPE вывода информации на экран дисплея, включая символ перевода строки (!), указатель формата вывода чисел (%A.0B), символ вывода на экран текущих значений всех переменных (⍰), символ трассировки (?). Все эти операнды выполняют те же функции, что и в операторе TYPE. Для экономии памяти запятую между открывающими или закрывающими кавычками и символом (!) или последующей переменной в обоих этих операторах можно опускать, т. е. писать

1.1 A ! «длина»L,! «ширина»B;T ! «площадь»L*B

При исполнении оператора ASK, если пользователь не намерен изменять значение переменной, следует вместо повторного ввода того же значения нажать клавишу ⓐ. Для отмены ввода ошибочно набранной последней цифры во вводимом числе следует нажать клавишу \ (обратная косая черта), а для отмены всего набранного на экране числа — клавишу **ЗБ —.**

Оператор DO является одним из наиболее эффективных программных средств Фокала. С его помощью можно вызвать на исполнение любую строку программы (в форме D NN.nn), любую группу строк (в форме D N), смоделировать оператор выбора (в форме D B, где B — переменная, имеющая значение номера строки или группы строк) и даже организовать исполнение любого заданного числа строк (в одной или нескольких группах). В последнем случае оператор применяется в форме D NN.nn, а в каждой исполняемой по этому оператору строке, кроме последней, в конец необходимо поставить оператор GOTO для передачи управления следующей строке. Применение этого оператора в сочетании с многооператорными строками позволяет резко уменьшить размеры программы пользователя.

Оператор FOR является оператором цикла с постусловием, поэтому при любом сочетании аргументов он выполняется хотя бы один раз. Следует также

учесть, что нормальная работа этого оператора происходит, если шаг приращения переменной цикла положителен, а начальное значение переменной цикла меньше ее конечного значения, знаки этих величин значения не имеют. Тело цикла может отсутствовать. В форме F I=A,E этот оператор может использоваться для программной организации паузы в программе. Для получения паузы на каждую секунду нужно предусмотреть около 400 пустых циклов.

Оператор IF условного перехода причиняет обычно наибольшие неприятности программистам. Он имеет следующий полный синтаксис:

I(X) N, Z, P

где X — арифметическое выражение, а N, Z, P — номера строк (или переменные, имеющие смысл номеров строк), которым передается управление при отрицательном, нулевом и положительном значении X соответственно. Отметим, кстати, что все пробелы здесь могут быть опущены. Главные сложности возникают при попытках организовать правильную передачу при нуле, если X получается не целым. Особенности машинной арифметики здесь легко могут привести к ошибкам. Одним из возможных выходов из положения является замена точного значения X его приближенным значением с ограниченным числом значащих цифр. Рассмотрим пример реализации этого приема:

```
2.2 T !«символ элемента:»;
S N=FCHR (FCHR(—1)); S B=N/23
2.3 I(FITR((B—3.13043)*1000))2.4,3.1
```

В этом фрагменте при вводе с клавиатуры символа H (код его равен 72) управление будет правильно передано строке 3.1. Если же записать строку 2.3 в виде

```
2.3 I(B—3.13043)2.4,3.1
```

то передачи управления на строку 3.1 выполнено не будет.

Если условия задачи позволяют, лучше избегать передачи управления оператором IF при X=0, а если это возможно, то стараться использовать для вычисления X целочисленную арифметику.

Оператор KILL применяется в программах редко, поэтому в руководстве пользователя он не описан. Он предназначен для сброса (блокировки прерываний) всех внешних устройств БК и идентичен по действию машинной команде RESET. Если вставить оператор KILL в программу, то после его выполнения будет запрещено прерывание программы от клавиатуры, поэтому ввести информацию по запросу оператора ASK или функции FCHR станет невозможно.

Операторы группы LIBRARY, предназначенные для записи программ и данных на магнитную ленту и чтения с ленты, описаны в руководстве пользователя подробно. Отметим только, что все переменные записываются на ленту оператором LIBRARY OUTPUT под своими именами. Запись производится отдельными блоками по 511 переменных в каждом блоке. Блоки нумеруются для служебных целей начиная с нулевого. Эта нумерация производится автоматически, причем номер блока выводится в 16-й позиции имени файла. При последующем вводе файла данных с магнитной ленты вводить эти номера при имени файла не нужно.

Отметим также, что при вводе файла данных с ленты по оператору LIBRARY INPUT стирания из памяти программы и введенных до этого переменных не происходит, но если на ленте окажутся переменные с теми же именами, что и хранящиеся в памяти, то последним будут присвоены новые значения.

Операторы группы PASS применяются для выхода из системы Фокала в мониторную систему диагностики (МСД). Поскольку работа в режимах МСД выходит за рамки Фокала и требует специального детального описания, мы этого вопроса касаться здесь не будем.

Несколько слов об операторе SET. С целью сокращения размера программ в этом операторе можно опускать число ноль, так что вместо S A=0 можно писать S A=. Вместо буквы S в этом операторе в некоторых программах встречается буква F, например: F B= =FSIN (X). Читатель может возразить, что F используется для обозначения оператора FOR, а не SET. Да, это так. Но если в операторе FOR оставить

только первый аргумент, опустив два остальных, то он превращается в обычный оператор присваивания, идентичный оператору SET. Такой прием, к сожалению, иногда используется, поэтому его следует знать — но не следует применять!

В операторе TYPE для сокращения размера программ можно употреблять символ перевода строки (!) вместо запятой, разделяющей текстовые сообщения, размещаемые на разных строках экрана, и опускать закрывающие кавычки, если они стоят в конце оператора, например:

T!«Первая строка»!«Вторая строка

Оператор XECUTE (или сокращенно X) применяется тогда, когда требуется исполнить какую-либо функцию или вычислить выражение, но вычисленные значения не нужно выводить на экран дисплея. Чаще всего он используется для выполнения управляющих команд, вводимых функцией FCHR. Для пользователей, программирующих для БК-0010 в машинных командах, будет интересно узнать, что с помощью этого оператора можно также легко переводить вещественные числа в их внутреннее представление. Для этого набирают в диалоговом режиме оператор X N, где N — целое или вещественное число, и нажимают клавишу **ВВОД**. Затем переходят в режим ТС самодиагностики МСД и выводят на экран содержимое ячеек памяти 1710, 1712 и 1714 для просмотра. В младшем байте первой ячейки будет храниться порядок числа N, во второй ячейке — старшие разряды его мантиссы, а в старшем байте третьей ячейки — младшие разряды мантиссы. Указанные ячейки называются флаком. При хранении числа в других ячейках памяти происходит небольшая трансформация этого представления, обусловленная тем, что вне флака под вещественное число отводится только две ячейки памяти: в первой ячейке в старшем байте хранятся младшие разряды мантиссы, а в младшем — порядок числа, во второй ячейке — старшие разряды мантиссы и знак числа.

Логические операции. Фокал БК не имеет специальных логических переменных и средств их обработки, однако их нетрудно сгенерировать программно.

Если условиться, что логические переменные A и B могут принимать только два значения: +1, соответствующее логическому значению «истина», и —1, соответствующее логическому значению «ложь», то основные логические операции с этими переменными можно определить в соответствии с табл. 2.

Таблица 2

Логическая операция	Символьная запись	Эквивалентная запись в Фокале
Отрицание	$\bar{A}$	FSGN (—A)
Сложение	$A \vee B$	FSGN (A+B+1)
Умножение	$A \wedge B$	FSGN (A+B—1)
Исключающее ИЛИ	$\overline{A \equiv B}$	FSGN (—(A*B))
Эквивалентность	$A \equiv B$	FSGN (A*B)
Импликация	$A \supset B$	FSGN ((A+B)— —2+B)

Эти операции можно соединять в сложные логические выражения, последовательность выполнения операций в которых следует устанавливать при помощи скобок.

Экономия памяти. При работе с ПЭВМ у каждого программиста рано или поздно на экране появляются пренеприятные сообщения ПЕРЕПОЛНЕНИЕ ПАМЯТИ ТЕКСТОМ ПРОГРАММ или НЕТ МЕСТА ДЛЯ ПЕРЕМЕННЫХ, которые в самый неподходящий момент напоминают, что память ПЭВМ не бесконечна. Чтобы в этом случае не отказываться от программы, в которую вложен немалый труд, можно попытаться сократить ее.

Разумеется, радикальным путем сокращения программы является использование более короткого алгоритма решения поставленной задачи, но это отдельный и далеко не простой вопрос. Рассмотрим те пути, которые не затрагивают существа алгоритма.

Наиболее мощным из них является использование подпрограмм, написанных в машинных командах. Для совмещения машинных подпрограмм с Фокалом можно использовать системные программы FOCOD, ИВФ, ИВФ МАССИВЫ и СФМ, описанные ниже.

Сокращайте имена операторов до од-

ной буквы, а операторы набирайте без пробелов между операндами. Для обозначения переменных по возможности также используйте однобуквенные имена.

Практикуйте многооператорные строки максимальной длины (до 79 символов).

Повторяющиеся последовательности операторов оформляйте при первом употреблении в виде либо отдельной строки, либо последовательности строк, соединенных операторами GOTO, либо отдельной группы строк, которые впоследствии вызывайте на исполнение операторами DO.

Для служебных переменных (переменных цикла, индексов, переключателей составного оператора выбора и т. п.) используйте в разных местах программы одни и те же имена, например I, J, K, L.

Если конечный результат является значением некоторого арифметического выражения и нигде в программе больше не употребляется, то экономнее не вычислять его, как обычно, в операторе SET, а включать целиком арифметическое выражение в оператор вывода TYPE. Так, например, из двух строк

S A=(FCOS(X))-2;T "A=", A

T "A=", (FCOS(X))-2

вторая экономнее первой на 14 байтов (6 за счет сокращения текста и 8 за счет отказа от ввода переменной A в память).

Сокращайте текст операторов ASK, TYPE и SET с помощью приемов, которые были описаны выше.

Хотя каждый из описанных здесь приемов в отдельности дает небольшую экономию, но, когда они все применены в большой программе, эффект будет значителен, причем он состоит не только в экономии памяти, но и в увеличении скорости исполнения программы.

Фокал часто упрекают в медленности исполнения его программ. С этим нельзя согласиться. Проверка на одинаковых вычислительных алгоритмах показывает, что программы на Фокале исполняются значительно быстрее, чем на Бейсике, причем иногда в несколько раз. Конечно, Фокал с его интерпретатором нельзя сравнивать с языками программирования, снабженными компиляторами или их подобиями, каким, например, снабжен Бейсик БК. Но если сравнение вести с интерпретаторами Бейсика, то преимущество Фокала в скорости неоспоримо.

Перейдем теперь к описанию расширений Фокала.

Сборка фокальных программ. Системная программа СБОРКА (автор — В. Т. Монахов) позволяет вводить с магнитной ленты программу на Фокале не целиком, как по оператору LIBRARY GET, а в виде отдельных блоков. Каждый такой блок кодируется и отлаживается самостоятельно; после окончания отладки и тестирования к нему добавляют пустую строку со старшим номером, т. е. набирают номер строки, превышающий все остальные номера строк в блоке, и нажимают **ВВОД.** После этого блок выводят на магнитную ленту обычным способом. При сборке фокальной программы из блоков вначале вводят в память программу СБОРКА, а затем с ее помощью поочередно вводят начиная с младших номеров строк, записанные на магнитной ленте блоки. СБОРКА автоматически подстыковывает каждый из них в конец формируемой в памяти программы и оформляет все необходимые связи. После ввода очередного блока можно вводить или редактировать любые строки программы с клавиатуры, только в конце этой работы нужно так же, как и при выводе блока на ленту, ввести в программу пустую строку со старшим номером. После этого можно продолжить ввод блоков с магнитофона. Завершив ввод блоков, программу СБОРКА можно исключить из собранной фокальной программы, если не хватает памяти, но можно и оставить в памяти, так как на работу программы она не влияет.

Возможность объединения фокальных блоков в единую программу позволяет резко расширить возможности БК и сократить время на разработку его прикладного программного обеспечения за счет широкого использования тематических библиотек стандартных блоков, из которых, как из кубиков конструктора, можно собирать интересующие пользователя программы.

Перенумерация строк. Полезным дополнением к программе СБОРКА яв-

ляется подпрограмма перенумерования строк фокальных программ (автор — В. Т. Монахов). Такая операция может потребоваться, если два блока имеют совпадающие номера строк или если при отладке программы оказалось, что нужно включить дополнительную строку, а сделать это без раздвижки строк нельзя. Параметрами подпрограммы перенумерации являются номер строки, с которого начинается изменение номеров, новый номер, присваиваемый этой строке, и шаг нумерации. Эта подпрограмма может объединяться с программой СБОРКА или входить в состав других системных программ, описанных ниже.

Машинные подпрограммы в Фокале. Для машинных подпрограмм, исполнение которых протекает без обмена данными между фокальной частью и подпрограммой, очень удобным инструментом связи является программа FOCOD (автор — В. Н. Андреев). Сущность ее состоит в том, что в Фокале вводится дополнительная функция FCHR (1), передающая управление из Фокала по абсолютному адресу 2100, начиная с которого пользователь получает возможность размещать любую подпрограмму в машинных командах. По завершении выполнения этой подпрограммы управление передается оператору фокальной программы, непосредственно следующему за оператором вызова FCHR (1). С Фокодом работают так. После ввода программы FOCOD программист переводит БК в режим директив отладки мониторной системы диагностики и далее с клавиатуры или с магнитной ленты вводит машинную подпрограмму с адреса 2100. Далее выполняются необходимые операции связи этой подпрограммы с Фокалом и командой 2000G БК переводится в режим Фокала. Теперь с клавиатуры можно вводить и редактировать текст фокальной части программы. По завершении отладки и тестирования объединенная машинно-фокальная программа может выводиться на магнитную ленту (и затем вводиться с нее) обычными средствами Фокала.

В принципе с помощью Фокода можно вводить не одну подпрограмму, а несколько, и не только без параметров, но и любым числом параметров. В этих случаях организацию переключения с одной подпрограммы на другую, а также передачу параметров из Фокала в подпрограмму и/или результатов из подпрограммы в Фокал приходится предусматривать программным путем, например с помощью функции FX.

Однако в таких случаях удобнее применять другую системную программу, например ИВФ (автор — А. П. Казанцев) или СФМ (автор — В. Т. Монахов). Программа ИВФ (интерфейс внешних функций) описана в [2].

Она позволяет включать в состав Фокала до 26 дополнительных функций произвольного назначения. ИВФ загружается в память ПЭВМ с магнитной ленты в режиме мониторной системы диагностики с адреса 20000. Далее в этом же режиме она дополняется необходимым числом подпрограмм вычислений дополнительных функций, под размещение которых отводится память с ячейки 20572. По окончании этой работы созданную машинную программу можно записать на магнитную ленту, если она потребуется в дальнейшем, а можно сразу перейти в режим Фокала, в котором с клавиатуры или магнитной ленты необходимо ввести фокальную часть программы. Под ее размещение в ИВФ отводится 7К байтов. Если необходимо создать отдельные массивы упакованных байтовых данных, а также целых и вещественных чисел, то можно воспользоваться реализацией программы ИВФ, названной ИВФ МАССИВЫ (автор — П. В. Полянский). В ней система ИВФ дополнена функциями, позволяющими зарезервировать память под указанные массивы, а также вводить и выводить из них данные.

Наиболее универсальным расширением Фокала является программа СФМ (символьные функции и массивы). Она включает все перечисленные выше расширения Фокала и, кроме того, вводит в него символьные переменные.

Программа СФМ объемом чуть более 2К написана в машинных командах и располагается в оперативной памяти БК между системной областью Фокала и текстом программы пользователя. Она скомпонована так, что составляет с про-

граммой пользователя единое целое и поэтому может выводиться на магнитную ленту и вводиться с ленты обычными операторами Фокала.

СФМ позволяет вводить символьные переменные и константы с клавиатуры и с магнитной ленты, сравнивать символьные величины между собой, вырезать из них фрагменты, модифицировать их с клавиатуры и программными средствами, определять их длину, вычислять числовое значение набора цифр и целых арифметических выражений, входящих в их состав, выводить символьные величины на экран, принтер или магнитную ленту, стирать их из памяти, производить поиск символьной величины в памяти по ее начальному фрагменту произвольной длины.

Кроме того, программа СФМ имеет функции пересчета десятичных чисел в восьмеричные и восьмеричных в десятичные с отображением результатов пересчета на экране, а также набор функций, обеспечивающих работу с двумя массивами упакованных чисел — целых и вещественных. Эти два массива, как и массив символьных величин, расположены вне области ОЗУ, доступной Фокалу, поэтому они не стираются оператором ERASE, что создает дополнительные удобства для пользователя, позволяет легко организовать глобальные и локальные переменные в программе.

Удобным отличием массивов программы СФМ от массива обычных переменных Фокала является одновременный их вывод на магнитную ленту при записи программы на магнитофон, хотя допустим и раздельный вывод.

Кроме 17 дополнительных функций, выполняющих 23 различные операции, программа СФМ предоставляет пользователю возможность запрограммировать в машинных командах еще 11 дополнительных функций любого назначения. После ввода программы в компьютер ее можно дополнять любой фокальной частью. Эту часть можно вводить не только с клавиатуры, но и с магнитной ленты, причем не обязательно целиком, а в виде подсоединяемых фокальных блоков. Введенную фокальную программу и ее отдельные группы строк можно перенумеровать, изменяя как номера строк, так и интервалы между строками.

Для программиста удобно и то, что в СФМ предусмотрена возможность свободного перехода из режима Фокала в режимы мониторной системы диагностики и обратно без стирания фокальной части программы.

Таким образом, Фокал, дополненный программой СФМ, по своим возможностям приближается к Бейсику БК, выгодно отличаясь от него компактностью и, главное, предоставлением пользователю большей памяти для размещения программы и данных.

Рассмотрим кратко дополнительные функции, вводимые программой СФМ. Все они имеют в названии две одинаковые первые буквы FF (от английского fyrther function), а третья буква отражает специфику функции и в какой-то мере тоже мнемонична.

Дополнительные функции можно использовать как обычные встроенные функции Фокала и в диалоговом, и в программном режиме. Их можно вкладывать друг в друга и использовать рекурсивно, если требуется. Неправильный синтаксис или ошбочное использование функций фиксируется системой диагностики ошибок.

Функция FFA — от ASK — предназначена для ввода символьной величины с клавиатуры. Вводимые величины не требуют предварительного объявления и могут содержать от 1 до 127 любых символов БК. Результатом исполнения функции является размещение введенной величины в массиве символьных переменных и выдача порядкового номера, присвоенного компьютером этой переменной. Переменные в массиве размещаются вплотную друг к другу, разделителем служит внутренний код клавиши **ВВОД.** При необходимости зарезервировать место под последующее наращивание переменной при вводе ее пользователь должен добавить в конце нужное число пробелов. Символьные переменные могут быть найдены в массиве по их порядковому номеру или по поисковому образу, представляющему начальный фрагмент достаточной для идентификации длины.

Функция FFB — от BROKEN — обеспечивает перенумерацию строк фокаль-

ной программы начиная с заданного старого номера строки путем присвоения ей нового заданного номера, а всем последующим строкам — больших номеров с заданным шагом нумерации.

Функция FFC — от COMPARE — предназначена для сравнения двух символьных переменных на равенство и неравенство с выдачей нуля при равенстве, +1, если больше, и —1, если меньше. Пробелы в конце переменной при сравнении не учитываются. Буквы русского алфавита для целей сравнения перекодируются в новый код, в котором значение кода буквы согласуется с ее порядковым номером в русском алфавите и все коды русских букв меньше, чем коды латинских букв. Эта функция не различает заглавные и строчные буквы, а сравнивает только содержание слов.

Функции FFD и FFO — от DECIMAL и OCTAL — предназначены для пересчета чисел, являющихся их аргументами, из восьмеричной системы счисления в десятичную и из десятичной в восьмеричную соответственно. Функция FFO одновременно выводит значение восьмеричного числа на экран.

Функция FFE(A) — от ERASE — в зависимости от знака аргумента A стирает либо массив символьных переменных, либо все массивы СФМ. Допустимо частичное стирание массива символьных переменных.

Функция FFF — от FIND — осуществляет поиск символьной переменной в массиве по ее поисковому образу. Последний может как задаваться с клавиатуры, так и находиться в памяти. Длина поискового образа может быть от одного до 30 символов. Если данному поисковому образу в массиве соответствует несколько переменных, то они все поочередно будут идентифицированы. Повторный набор поискового образа при этом не требуется.

Функция FFI — от INTEGER — обеспечивает работу с массивом упакованных целых чисел, в котором каждое число занимает одно машинное слово и идентифицируется по номеру. С помощью функции FFI можно в любом порядке вводить числа в массив и выводить из него. Кроме того, она позволяет выделять целую часть смешанных чисел. В отличие от стандартной функции Фокала FITR функция FFI действует не методом отбрасывания дробной части, а действительно выделяет целую часть, как это делает, например, функция INT Бейсика БК.

Функция FFL — от LENGTH — определяет длину символьной переменной, включая пробелы в ее конце.

Функция FFM — от MODIFY — предназначена для модификации символьных переменных путем замены их целиком или частично другим набором символов. Модифицирующий текст является либо непосредственным аргументом функции, либо символьной переменной, уже имеющейся в массиве. В обоих случаях в память БК вводится только столько символом из этого текста, сколько позволяет длина модифицируемой переменной.

Функция FFN — от NUMBER — служит для вычисления числового значения непрерывной последовательности цифр, встретившейся первой в тексте заданной символьной переменной.

Функция FFR — от REAL — обеспечивает работу с массивом упакованных вещественных чисел, в котором каждое число занимает два машинных слова. Ее возможности аналогичны возможностям функции FFI.

Функция FFS — от SEGMENT — выполняет вырезку из указанной символьной переменной заданного фрагмента и размещение его в виде новой символьной переменной в памяти БК.

Функция FFT — от TYPE — обеспечивает вывод на экран значений нескольких символьных переменных в порядке перечисления их имен в списке аргументов.

Функция FFV — от VACATE — резервирует память для размещения массива символьных переменных (необходимый объем памяти задается общей длиной массива) и двух массивов упакованных чисел (необходимый объем задается числом чисел в каждом из массивов). Эта функция не стирает уже введенных символьных величин и поэтому может применяться многократно, например, для увеличения размера символьного массива.

Функция FFX — от команды

XECUTE — вычисляет значение арифметического выражения, входящего в состав символьной переменной. В составе этого выражения допускается использование переменных Фокала, встроенных и дополнительных функций, скобок.

Кроме того, в СФМ имеется подпрограмма сборки фокальной программы из отдельных присоединяемых блоков, записанных на магнитной ленте.

СФМ допускает работу в любом режиме экрана, в том числе в режиме расширенной памяти.

Из этого краткого обзора видно, что дополнительные функции СФМ серьезно расширяют возможности не только самого Фокала, но вообще компьютеров БК. Дело в том, что отмеченные в начале статьи недостатки Фокала и Бейсика БК не позволяли создавать сколько-нибудь значительные по объему учебные и информационно-справочные программы, например базы данных, словари и справочники, не прибегая к оверлейным структурам и хранению части данных на ленте. Это увеличивало время доступа к данным, сводя на нет преимущества вычислительной техники.

Применение СФМ позволяет резко сократить объем программ и тем самым высвободить память для размещения данных, организованных к тому же в предельно упакованном виде. Примером справедливости этих слов служит программа ARRAS (автор — В. Т. Монахов) англо-русского и русско-английского словаря, которая позволяет разместить в оперативной памяти БК одновременно два словаря, каждый из которых может состоять не менее чем из 1000 словарных статей. Вот полный текст программы (без словарного массива) этого словаря, который без лишних слов показывает достоинства разработки:

```
1.1 T!«Введи поисковое слово: »;S A=;S B=
1.2 S C=FFF(A,B);S R=; I(—FX(0,4212, —256))1.3;S R=1
1.3 T!;X FFT(C);T " "; I(R)R,1.4;X FFT(C—1);G 1.5
1.4 X FFT(C+1)
1.5 I (FCHR(—1)—13)1.6,1.1
1.6 S A=1; S B=C+2;G 1.2
2.1 T!«Введи новое слово»!«английское:»;T FFA( );T!«русское:»;T FFA( );G2.1
```

Несмотря на неправдоподобную краткость, эта программа обеспечивает поиск словарной статьи и печать ее на экране по поисковому слову любой длины, хоть из одной буквы, автоматическое переключение словаря с англо-русского на русско-английский или наоборот, в зависимости от того, какими буквами набрано поисковое слово. Простое нажатие пробела вызывает на экран вторую статью на тот же поисковый образ, и так может продолжаться до тех пор, пока не будут исчерпаны все словарные статьи, начинающиеся с заданного поискового образа. Нажатие клавиши **ВВОД** прекращает этот поиск и возвращает программу к строке запроса на ввод нового поискового образа с клавиатуры.

В программе имеется группа строк 2, которая может быть использована для ввода в словарь новых словарных статей, если еще не исчерпан объем ОЗУ. Ввод новых статей закольцован, выход из него осуществляется клавишей **ВВОД.** Как видим, программа достаточно универсальна, хотя работа с ней предельно проста и доступна даже школьникам младших классов.

В заключение следует отметить, что выбор дополнительных функций программы СФМ был обусловлен намерением автора приспособить Фокал для эффективной работы с текстовой информацией и облегчить разработку фокальных программ. С таким же успехом СФМ может быть приспособлена для выполнения других задач, например для создания высокодинамичных игровых программ или решения задач автоматизации контроля и управления объектами.

Литература

1. *Зальцман Ю.* Как засеять двухмерное поле // Наука и жизнь, 1987, № 10.

2. *Казанцев А. П.* Интерфейс внешних функций интерпретатора Фокал БК-0010 // Микропроцессорные средства и системы. 1987. № 4.

Ю. КАДЕЕВ

Язык программирования «Илюша»

Понятно. Если понятно, то хорошо, иначе не хорошо.

Приведенные высказывания не напоминают фразы на каком-либо языке программирования, особенно если учесть правильно расставленные знаки препинания. Однако эти фразы синтаксически верны на языке программирования «Илюша». Добавим к ним третью: «Хорошо?» (тут и в дальнейшем при необходимости кавычками будут выделяться фрагменты языка). При пуске программы, состоящей из указанных трех фраз, на экране появится сообщение «Да». Если первую фразу переписать так: «Не понятно», то будет выдано сообщение «Нет». Если же первую фразу вообще убрать из текста программы, то появится сообщение «Не знаю».

В язык «Илюша» не введен механизм логического вывода, имеющийся в языках Лисп, Пролог. Возможность написания таких фраз появляется при введении в язык понятия утверждения. Под утверждением понимается фраза, состоящая из двух элементов — числового идентификатора и конца фразы — или из трех — «Не», числового индентификатора и конца фразы. Допускается частицу «Не» заменять на «!». Конец фразы — точка либо точка с запятой. Таким образом, фраза «Понятно.» есть утверждение в языке «Илюша». Интерпретатор, выполняя часть программы, состоящую из этой фразы, присваивает числовой переменной «понятно» значение 1. При выполнении фразы «Не понятно.» числовой переменной присваивается значение 0. Далее интерпретатор проверяет значение выражения, стоящего после «Если», и, если оно положительно, осуществляет переход к части программы после «то», в противном случае — к части программы, стоящей после слова «иначе». Команда «Хорошо?» заставляет интерпретатор вывести на экран значение переменной «Хорошо», при этом в одном из режимов единица заменяется на «Да», ноль на «Нет» и остальные значения на «Не знаю». Таким образом, диалог с ЭВМ в некоторых простых случаях приближается к обычному диалогу между людьми.

Возникает вопрос: а нужно ли это? Ситуация, когда язык общения с ЭВМ изучается каждым будущим работником общества, возникла впервые. Трудно ожидать, что имевшиеся до этого средства и способы обучения сработают так же, как и раньше, когда языками общения с ЭВМ владела лишь узкая группа профессионалов. Собственно, уже имеющийся опыт преподавания информатики в школе подтверждает, что наиболее трудным для изучения оказываются темы, связанные с программированием. Жизнь заставляет отбросить или существенно изменить подходы к оценке тех или иных учебных средств, имеющих отношение к информатике вообще и к программированию в частности.

Информатика, и это одно из основных ее отличий от остальных школьных предметов, станет в ближайшее время самым социально значимым школьным предметом. В наше время умение составлять алгоритмы становится наиболее приоритетным, престижным и одним из самых высокооплачиваемых. Следовательно, учебное обеспечение информатики должно делать ее равнодоступной.

Когда мы говорим о языке программирования, то нужно иметь в виду и то, что это новый для данного ученика язык общения. Если учесть, что в скором времени изучение информатики будет происходить в младших классах, то этот факт таит в себе потенциальные трудности, возникающие при параллельном изучении двух языков с одинаковой лексикой, но разными синтаксисами. Изучение русского языка в русских школах заканчивается к VIII классу. В VI, VII, а тем более в начальных классах навыки грамматики языка или не сформировались, или не устоялись. Если ученику V класса предложить написать фразу IF A=2 THEN A=1, то он не спросит, почему перед THEN нет запятой, так как у него в данный момент нет ассоциации с правилами написания запятой на русском языке. Но как быть этому же ученику, если он регулярно вынужден будет писать фразу типа ЕСЛИ A=2 ТО A=1? Причем на попытку поставить запятую в нужном месте ЭВМ будет откликаться сообщением об ошибке. Ясно, что в этом случае изучение как русского, так и языка программирования будет затруднено. Уже из приведенного примера следует, что язык программирования с русской лексикой, предназначенный для массового изучения, не должен противоречить нормам, правилам, а желательно и традициям русского письменного и устного языка. Выполнение этого требования было одной из основных посылок при создании «Илюши».

В его основу был положен и еще один принцип. Известно, что при планировании городских парков используются два подхода. В одном случае все тропы планируются заранее; другой подход основан на том, что вначале люди протаптывают дорожки, исходя из своих интересов, а затем эти дорожки и тропинки утверждаются на плане и асфальтируются.

Второй подход и был использован при создании синтаксиса «Илюши». В качестве полигона использовались уже имеющиеся языки: Бейсик, Фокал, алгоритмическая нотация.

В кружковой работе, на факультативе, а также (частично) на уроках ученикам разрешалось писать отдельные конструкции не как положено, а так, как им нравилось или было удобнее. Кроме того, были проанализированы ошибки учеников при выполнении контрольных работ. В некоторых случаях выяснилось, что имеется достоверная закономерность в ошибках, а также в стремлении учеников сделать синтаксис более удобным.

Пример программы на «Илюше».

```
Начало.
    К=12,5.
    К=К*2.
    К ?
Конец.
```

Переменная К примет значение 25, оно будет выведено на экран. Десятичные числа записываются через привычную запятую, а окончание предложения отмечается точкой, что не противоречит письменному русскому языку. «Начало.», воспринимается интерпретатором как комментарий и служит для увеличения наглядности. По команде «Конец.» интерпретатор прекращает дальнейшее выполнение программы.

Числовое выражение «К*2» можно было бы записать и иначе: 2К. Интерпретатор понимает привычное здесь для ученика умолчание. Вообще в язык встроено большинство школьных умолчаний. В выражениях «2(К+М)», «К(К+М)», «(К1+М1)(К2++М2)» интерпретатор «видит» знак умножения.

В языке имеется два типа чисел: десятичные и дроби. Известно, что на долю описателей переменных у школьников падает немалое число ошибок, а главное, неясна их роль в программе. В сущности, описание переменных необходимо создателям транслятора, а не пользователям. В языке «Илюша» предварительное описание переменных не обязательно.

Существует точка зрения, что ученик должен знать способы внутреннего представления чисел в ЭВМ. Считается даже, что этот факт имеет основополагающее значение. Другое похожее мнение, сближающее информатику с системным программированием, состоит в том, что команда присваивания является одной из основных. Однако развитие информатики опровергает это: достаточно упомянуть о проекте ЭВМ пятого поколения. От пользователей таких машин (и даже от программистов) не потребуется знание ни того, ни другого. Уже сейчас имеется немало языков программирования, не требующих понимания внутреннего представления чисел в ЭВМ и осуществляющих присваивание косвенно.

Очевидно, что развитие информатики идет в направлении приспособления ЭВМ к человеку и его мышлению, а не наоборот. Думается, что программа, учебники и учебные языки программирования должны учитывать это и по возможности несколько опережать ситуацию.

Если интерпретатор языка «Илюша» встречает неописанную переменную, он пытается «понять» ее тип из контекста. Если это не удается, он вступает в диалог с пользователем с целью выяснения типа этой переменной. В последней программе тип переменной К очевиден для интерпретатора из текста программы.


```
Начало.
    К=2/5.
    Л=3/8.
    (4+Л) ?
    ((К+Л):2) ?
Конец.
```

В этой програме переменным К и Л присваиваются не результаты деления 2/5 и 3/8 соответственно, а именно дроби 2/5 и 3/8. Переменные К и Л определяются самим интерпретатором как переменные типа дробь. Знак / не является знаком для обозначения деления, а лишь, как это имеет место в курсе математики средней школы, для обозначения дроби. Для обозначения деления используется привычное двоеточие. По команде «(К+Л) ?» будет выведена на экран дробь 31/40, а по следующей команде — дробь 31/80. Таким образом, язык можно использовать как калькулятор не только для десятичных чисел, но и для дробей. Для них существуют встроенные функции: сокращение дроби, выделение знаменателя и числителя, нахождение наименьшего общего знаменателя, что позволяет использовать язык на уроках математики в младших классах.

```
Начало.
    К=100/101.
    Л=101/102.
    Максимум=К\/Л.
    Максимум ?
Конец.
```

Индентификаторы переменных не ограничены по длине, что позволяет тексты программ, предназначенных для демонстрации, делать более наглядными. В данной программе определяется максимум из двух дробей К и Л, затем он распечатывается. Знак «\/» используется для обозначения наибольшего из двух чисел. По функции МКС с неограниченным числом аргументов можно находить наибольшее из нескольких чисел. Аналогично действуют знаки «/\» и функция МНМ.

```
Начало.
    1→К.
    12→Л.
    Пока (К<15) и (К<Л) выполнять [
      К ?
      К=К+1.
    ]
Конец.
```


В этой программе выводятся на экран все натуральные числа, меньшие 15 и Л, равного 12, т. е. от 1 до 11. Для присваивания значений переменным К и Л здесь используется иная конструкция. Каждый пользователь может выбрать то, что ему нравится.

В фрагменте «(К<15) и (К<Л)» мы имеем дело с конъюнкцией двух выражений. При конъюнкции каждое числовое выражение считается истинным, если его значение положительно, и ложным в противном случае. В качестве указания на конъюнкцию в языке разрешается использовать все те слова, которые употребляются для этой же цели в разговорной речи: «и», «а», «но», что соответствует прогрессивному принципу «как говорим, так и пишем».

```
Начало.
    1→К.
    12→Л.
    Пока К<20 выполнять [
      Если К>Л, но<18, то К ?
      иначе
      К=К+1.
      ]
Конец.
```

Будут выведены натуральные числа от 13 до 18. Фрагмент программы «Если К>Л, но<18» полностью соответствует разговорному языку. В качестве одного из критериев «понимаемости» учебного языка программирования можно предложить возможность осмысленного восприятия программ на этом языке на слух лицами, не знакомыми с данным языком. Ведь известно, что человек гораздо лучше говорит, чем пишет. Этому критерию указанная фраза вполне удовлетворяет.

Как уже было отмечено, интерпретатор языка совмещен с редактором. При создании текста редактор не только уточняет в случае необходимости тип переменной, но и постоянно следит за синтаксической правильностью текста. То, что язык «Илюша» приближен к русскому языку, имеет не только плюсы, но и потенциальные минусы. Ученику неизвестна грань между разрешенными и запрещенными в языке конструкциями, и имеется опасность его постоянного скатывания на рельсы «свободного» общения с машиной. В действительности этого не произойдет. Допустим, что в последней программе ученик попытается написать фразу «Пока К еще<20». Редактор не даст ему этого сделать. Как только будет набрана первая буква «е» в этой фразе, редактор ее выведет на экран, но курсор не перейдет на следующую позицию, а останется под этой буквой, в специальном же окне появится сообщение, что фраза непонятна. После этого можно заменить набранный символ на другой, вернуться назад, если станет ясно, что причина ошибки возникла раньше, и произвести необходимые изменения. Можно продвигаться и вперед по тексту, если причина ошибки обнаружится впереди. Нельзя только начинать новую самостоятельную фразу, не исправив или не удалив непонятный для редактора фрагмент.

Таким образом, ученик узнает об ошибке в момент ее совершения, а не спустя некоторое время, как это бывает в иных системах, причем сама ошибка так и не совершается. Это увеличивает скорость обучения.

Говоря об активности редактора «Илюши», следует упомянуть и его готовность в любой момент выдать справку по языку. Для этого нужно запросить помощь нажатием определенной клавиши. Весь набранный текст и положение курсора запоминаются, и редактор выдает меню, отражающее основные разделы языка. Если ученик забыл, как выводится информация на экран, то он выберет строку «Вывод информации на экран». После этого ему будет предложено новое, более детализированное меню. Выбрав соответствующий фрагмент, он получит необходимую информацию. Если при этом он забудет, как возвратиться к набору программы, он нажмет все ту же клавишу запроса помощи, и редактор подскажет, что ему нужно сделать.

Работа учителя в классе, оснащенном ЭВМ, напоминает работу центрального процессора в многотерминальной сети. Однако ясно, что никакой учитель с этой ролью не справится. Ученики вынуждены в ожидании ответа учителя на свой вопрос либо постоянно теребить его, либо действовать методом проб. Редактор «Илюши» в достаточной мере снимает эту проблему.

```
Начало.
    Ученик=«компьютеризация».
    Для К=2, 4, 8, 9, 12, 13, 14, 15 выполнять [
      Операция=Операция\/ВРЗ [Ученик, К].
    ]
    Операция ?
Конец.
```

В этой программе, как и во всех предыдущих, интерпретатор устанавливает типы переменных исходя из контекста. Текстовой переменной «Ученик» будет присвоено значение «компьютеризация». Затем выполняется списковый цикл «Для». Переменная К принимает последовательно значения, указанные в списке. Встретив идентификатор «Операция», интерпретатор сначала выясняет, что его тип — текстовый, и тут же присваивает этой переменной значение пустой строки. Затем восемь раз выполняет операцию объединений (склейки) строк. ВРЗ — функция, возвращающая тот символ переменной «Ученик», номер которого в значении этой переменной начиная с первого равен натуральному значению числа К. После выполнения цикла «Для» значение переменной «Операция» становится равным строке «операция». Затем на экран выводится слово «операция» — значение переменной «Операция».

Список встроенных функций языка «Илюша», работающих с текстовыми и числовыми переменными, несколько шире аналогичного списка языка Бейсик в стандарте MSX.

Помимо перечисленных редактор языка «Илюша» несет еще одну нагрузку. Предусмотрен отладочный режим работы интерпретатора. В нем используется три окна, размеры которых можно менять. В одном окне выводятся значения заказанных переменных в виде таблицы. В другом высвечивается листинг программы, причем курсор всегда стоит за последней выполненной командой. Продвижение листинга происходит автоматически или по командам. В третьем ученик дает команды отладчику, высвечивается стек обращения к подпрограммам, а также выводится другая информация интерпретатора.

В отладочном режиме можно изменить значение любой переменной, изменить последовательность прохождения программы, заказать пошаговый режим, причем один шаг может быть как полной командой, так и прохождением очередного символа. Предусмотрены и другие стандартные отладочные механизмы.

В режиме набора текста программы можно просматривать на экране одновременно два фрагмента программы.

В языке «Илюша» имеются средства обработки графической информации.

Любая графическая информация хранится в памяти в виде векторной фигуры, под которой подразумевается вектор или направленная ломаная. Графические переменные могут принимать значения векторных фигур. Все векторные фигуры виртуальны. При выводе значения графической переменной на экран используются специальные команды, в которых указывается, в какое место экрана и каким цветом выводить фигуру.

Имеются действия и функции над графическими переменными, причем все необходимые преобразования происходят вртуально. Например, можно переменной «Треуг» присвоить значение какого-либо треугольника (это делается указанием его координат), затем над этой переменной произвести гомотетию и присвоить полученное значение переменной «Треуг1». Затем можно значения обеих переменных вывести на экран в любое указанное место, повернув при желании один из треугольников.

Имеются также и стандартные опарации вывода графической информации на экран: вывод точки, вывод луча, проведение линии определенной толщины и цвета, вывод круга, окружности и эллипса. Каждое множество, выведенное на экран таким образом, может быть записано в память в виде векторной фигуры, присвоено графической переменной и затем подвергнуто любым преобразованиям. При наличии аппаратной поддержки в качестве графических констант, над которыми могут производиться любые преобразования, могут быть использованы любые выводимые на экран символы.

Таким образом, ученик, работая с интерпретатором языка, имеет дело с приближенными к школьной математике графикой и числовыми выражениями, пользуется лексическими и синтаксическими конструкциями, близкими к естественному языку, имеет возможность в любой момент запросить помощь и справку у интерпретатора, постоянно находится под контролем редактора, не дающего возможность совершить языковую ошибку.

В заключение отметим, что осуществление реализации довольно сложного синтаксиса языка «Илюша» стало возможно благодаря наличию в операционной системе ДЕМОС, в которой производилась работа над интерпретатором, программ синтаксического и лексического анализа YASS и LEX. Вообще, опыт работы в ОС ДЕМОС показывает ее хорошую приспособленность для создания педагогических программных средств.

Реализации интерпретатора существенно помогла приятная атмосфера активной поддержки гостей, которая характерна для кафедры прикладной математики ИПК Минавтопрома, где проходили работы по созданию интерпретатора.

Для получения интерпретатора достаточно выслать магнитный носитель по адресу: 440047, Пенза, ул. Минская, д. 1, кв. 231. Кадееву Юрию Ивановичу.

В. МАНЗЮРА, Д. ШМЕЛЕВ

Система для учебного моделирования языковых явлений

В настоящее время в лингвистике наряду с традиционными методами изучения языка все большее применение находит моделирование. Внедрение микрокомпьютеров позволяет использовать машинное моделирование не только в исследовательской, но и в учебной работе — при изучении теории и овладении практикой языка как в школе, так и в вузе.

Главное дидактическое достоинство моделирования состоит в том, что оно предполагает активное, осмысленное исследование материала, а не простое заучивание суммы готовых сведений о нем. Моделирование прививает обучаемым исследовательские навыки, у них заметно повышается интерес к предмету. Работа с компьютерной моделирующей системой приучает к активному управлению машиной, закладывает основы алгоритмического мышления на материале конкретной учебной дисциплины и способствует более легкому освоению в дальнейшем информатики и языков программирования.

Моделирование удобно применять при изучении открытых систем, разновидностью которых является естественный язык, а также его частные подсистемы. Применение вычислительной техники значительно расширяет возможности этого метода.

Нами разработана система для моделирования языковых явлений, которая называется СИМВОЛ. Система реализована на символьном языке микро-ЭВМ «Искра-1256», близком к Бейсику, но она легко может быть перенесена на другие компьютеры, более распространенные в обучении.

Назначение системы

Название описываемой системы представляет собой аббревиатуру, расшифровка которой может до некоторой степени прояснить ее назначение: Система Моделирования Выводимых Объектов Лингвистики. Под «выводимыми объектами лингвистики» понимаются языковые единицы (слова, морфемы, комплексы морфем и т. п.), которые могут быть построены («выведены») по определенным формальным правилам из других языковых единиц. Так, производимые слова выводимы из производящих, одни формы слов выводимы из других, в отношениях выводимости находятся восходящие к одному источнику слова разных родственных языков или разных диалектов языка, а также слово одной эпохи развития языка к этому же слову, взятому в другую эпоху.

Таким образом, система СИМВОЛ может быть использована в широком диапазоне: в работах по фонологии, словообразованию, морфологии, орфографии, сравнительной грамматике и лексикологии, истории языка и диалектологии.

СИМВОЛ может применяться в исследовательских и учебно-исследовательских работах (при подготовке курсовых и дипломных сочинений, рефератов), а также при обучении языку. Простота работы с системой делает ее доступной и для школьников средних классов, которые могут моделировать с ее помощью изучаемые по школьной программе правила словообразования, морфологии и орфографии. Старшеклассники способны создавать, более сложные модели, которые найдут применение при обобщающем повторении изученного материала. Студенты-филологи могут использовать моделирование при изучении различных дисциплин лингвистического цикла. СИМВОЛ может быть применен и на занятиях по практическому русскому языку с иностранными учащимися, которым моделирование поможет лучше усвоить чередования, «восстановить» нейтральное слово по уменьшительной форме, закрепить некоторые особенности русского правописания.

Методика моделирования

Моделирование языковых явлений с применением системы СИМВОЛ удобно тем, что не требует знания специального языка и техники программирования. Моделирование происходит в несколько этапов.

1. **Анализ материала.** Для построения модели необходимо иметь некоторые представления о структуре изучаемого объекта. Это требует обращения к материалу, в котором исследователь обнаруживает определенные формальные соотношения и закономерности. Эти закономерности на первоначальном этапе исследования могут быть сформулированы в произвольной форме. Иногда с помощью моделирования проверяются уже известные правила. В этих случаях этап первоначального анализа материала опускается.

Следующие этапы моделирования связаны с постепенной формализацией правил, приведением их к такому виду, с которым может работать компьютер.

2. **Анализ правил. Выявление условий.** Правила перехода от исходной формы к выводимой могут различаться наличием или отсутствием условий, в которых они реализуются. Так, при выведении современного русского слова (в его орфографическом облике) из древнерусского всякий Ѣ заменяется на Е — независимо от окружения, положения в слове и т. п. (ЛѢТО → ЛЕТО, О ЛѢТѢ → О ЛЕТЕ, ѢДУ → ЕДУ). Это правило не имеет условий.

Многие правила реализуются только в определенных условиях. Так, буква Ъ в русском письме употребляется после приставок на согласную перед буквами Е, Ё, Ю, Я. Условия такого типа можно назвать *контекстными*. При анализе правил, выполняющихся в контекстных условиях, эти условия должны быть подробно расписаны. В приведенном примере необходимо выявить полный список приставок, оканчивающихся на согласную.

Правила образования слов и форм также можно считать контекстными. Так, для построения увеличительной формы необходимо заменить А на конце существительного (это и есть условие) сегментом ИЩА (ЛАПА — ЛАПИЩА, ЖАРА — ЖАРИЩА и т. п.). При построении формы 1-го л. ед. ч. от инфинитива конечное сочетание ЗИТЬ заменяется на ЖУ (ВОЗИТЬ — ВОЖУ), СИТЬ — на ШУ (КОСИТЬ — КОШУ) и т. п.

В некоторых случаях реализация правила зависит от семантических условий. Таково, например, правило о приставках ПРЕ- и ПРИ-. Правила такого типа непосредственно не могут моделироваться на ЭВМ. Если же машинное моделирование семантических правил все-таки сочтено целесообразным, то их условия должны быть специальным образом помечены во вводимых формах, например: значение присоединения — цифрой 1, значение «очень» — цифрой 2, а запись будет выглядеть так: 1 — ПР? ШКОЛЬНЫЙ; 2 — ПР? ДОБРЫЙ. Иными словами, все семантические условия при машинном моделировании должны быть представлены как контекстные.

3. **Формализация правил и введение обозначений.** После того как правила и их условия выявлены, можно приступать к построению формальной модели, предназначенной для ввода в компьютер. Безусловные правила записываются в виде формулы X→Y, где X — заменяемый символ или цепочка символов, а Y — заменяющий. Например: Ѣ→Е.

В виде такой же формулы могут быть записаны и многие условные правила. Таковы, например, правила перехода от инфинитива к форме 1-го л. ед. ч., требующие замены сегментов на конце слова, который обозначается символом #:

1. ДИТЬ# → ЖУ# (ХОДИТЬ → ХОЖУ)
2. ЗИТЬ# → ЖУ# (ВОЗИТЬ → ВОЖУ)
3. СИТЬ# → ШУ# (КОСИТЬ→КОШУ)
4. ТИТЬ# → ЧУ# (КАТИТЬ→КАЧУ)
5. БИТЬ# → БЛЮ# (ЛЮБИТЬ→ЛЮБЛЮ)
6. ВИТЬ# → ВЛЮ# (ГОТОВИТЬ→ ГОТОВЛЮ)
7. МИТЬ# → МЛЮ# (КОРМИТЬ→КОРМЛЮ)
8. ПИТЬ#→ПЛЮ# (ЛЕПИТЬ → ЛЕПЛЮ)
9. ФИТЬ#→ФЛЮ# (ГРАФИТЬ → ГРАФЛЮ)

Правила 5—9 можно обощить: после губной согласной (Б, В, М, П, Ф) конечное ИТЬ заменяется на ЛЮ. Для этого необходимо ввести для всех губных обозначение, например символ + . Тогда вместо правил 5—9 в модели будет лишь одно правило:

+ИТЬ# → ЛЮ#

Однако при этом модель кроме правил должна включать в себя и «расшифровку» обозначений, в данном случае:

+ : БВМПФ

Использование обозначений позволяет делать модели более компактными и адекватными их языковой основе.

Обозначение должно состоять из одного символа, который не используется в записи материала и выводимого результата. В качестве обозначений можно использо-

---

Примечание. Символ # принят в лингвистике для обозначения границ слова. В этой же функции введен в СИМВОЛ.

вать цифры; при работе с материалом, записанным русскими буквами, удобны обозначения из латинских букв, не совпадающих по начертанию с русскими; можно пользоваться небуквенными знаками: +, —, ¤ и т. п. Нельзя употреблять в качестве обозначения знак #, которым в СИМВОЛе, согласно лингвистической традиции, обозначается левая или правая граница слова, а также точку, которая используется как команда для ЭВМ. Не рекомендуется в качестве обозначения применять знак' (апостроф), который может понадобиться для передачи ударения, мягкости согласных или как замена твердого знака. При моделировании орфографических правил бывает необходим символ для пропущенной буквы. Им может быть вопросительный знак или звездочка (*), и этот символ уже не может фигурировать в качестве обозначения, как и другие добавочные символы, используемые в записи материала.

Приводим примеры обозначений:

J : ЕЮЯ (все йотованные буквы);

W : ЖЧШЩ (все шипящие);

1 : АЕИОУЫЭЮЯ (все гласные);

— : КГХ (все заднеязычные согласные) и т. п.

Если условие, при котором выполняется правило, имеет специальное обозначение, то правило записывается следующим образом. Условное обозначение пишется на том же месте, где находится обозначаемая им буква в исходном слове, т. е. слева или справа от заменяемой буквы или комплекса букв. После стрелки, означающей команду «заменить на...», условное обозначение не повторяется. При замене обозначаемая им буква сохраняется в вывыводимой форме.

Допустим, надо записать правило: «После шипящей заменить О на Е». Обозначим шипящие символом W. Запись должна иметь следующий вид:

WO → Е.

Предположим, правило гласит: «Перед Е, Ю, Я заменить пропущенную конечную согласную приставки на сочетание З'». Обозначим гласные Е, Ю, Я символом J. Пропущенную букву будем передавать через * . Формальная запись выглядит следующим образом:

*J→ З'(РА ЕЗД→РАЗ'ЕЗД)

Иногда условие не исчерпывается указанием одного символа, стоящего до или после заменяемого сегмента. Тогда в формуле можно указать несколько обозначений подряд. Например, беглая гласная морфонема Е/∅ реализуется перед сочетанием согласных как Е. Введем для согласных обозначение +, а беглую морфонему передадим как ¤ . Правило будет иметь следующий вид:

¤++ → Е (МЕТ¤ЛКА → МЕТЕЛКА)

Возможны случаи, когда обозначения ставятся с обеих сторон заменяемого сегмента. Так, та же беглая Е/∅ в положении после гласной и перед согласной реализуется на письме буквой Й. Введя для гласных обозначение 1, а для согласных +, получим следующую запись:

1¤+ → Й (МА¤КА → МАЙКА)

При окончательном формировании модели вначале записываются обозначения, а затем правила. Одно обозначение можно использовать не более чем для 31 символа. Всего в компьютер можно ввести не более 20 обозначений.

Сложное лингвистическое явление невозможно описать одним правилом.

Поэтому несколько формул (правил) могут быть объединены в упорядоченный список. Максимальная длина этого списка в системе СИМВОЛ, реализуемой на ЭВМ «Искра-1256», составляет 120 формул. Выполнение списка заключается в последовательном применении заданных в нем правил к исследуемому материалу. При этом очень важным является порядок правил, который обычно отражает их последовательность в истории языка или иерархию в его современном состоянии.

При определенном навыке работы с моделирующей системой этап анализа правил сливается с этапом их формализации и построения модели.

4. Подбор материала для проверки модели. Одним из главных преимуществ машинного моделирования является возможность быстрой проверки полноты и правильности модели на большом материале. Это особенно ценно для работ исследовательского характера. Количество единиц исходного материала в СИМВОЛе ничем не ограничено, хотя за один раз можно обработать не более 30 единиц. Материал может быть записан на магнитную ленту и многократно обработан по разным моделям или разным вариантам одной модели. Результат моделирования также можно записать на магнитную ленту и использовать для дальнейшей обработки.

При учебном моделировании обычно нет необходимости проверять модель на всем материале. Здесь достаточно ограничиться примерами. При традиционных методах обучения закрепление изученного проводится на примерах, подобранных учителем или составителем учебной книги. При моделировании ученик должен, пользуясь словарями и текстами, сам подобрать примеры на все

случаи изучаемого явления, а иногда и сконструировать их, опираясь на собственную языковую интуицию. Это повышает активность учащегося и способствует лучшему осознанию изучаемого материала. Иногда удается обнаружить неполноту или неточность известного правила, найти неописанные исключения.

5. **Работа с ЭВМ.** Этот этап моделирования подробно описан в правилах пользования системой.

Перед тем как начать работу с компьютером, желательно уже подготовить модель и список слов (или других языковых единиц), которые следует проанализировать.

6. **Анализ результатов.** Это очень ответственный этап моделирования. Результаты, выданные машиной, сравниваются с реально представленными языковыми единицами. Случаи несоответствий анализируются. Иногда неправильные формы бывают образованы машиной в результате неточности или неполноты модели. Очень распространенная ошибка — неверный порядок правил. В таких случаях в модель вносятся необходимые уточнения и изменения (для этого в СИМВОЛе предусмотрено редактирование модели и словника), после чего счет на ЭВМ должен быть повторен. Среди неправильно образованных машиной форм могут быть и такие, которые представляют собой исключение в самом языке. Обнаруженные исключения можно ввести в модель.

Иногда при анализе результатов моделирования обнаруживаются закономерности, которые ранее не были замечены, конструируются потенциальные слова или формы. В этом состоит особая познавательная ценность метода моделирования. Например, удается установить, что в прилагательных с основой на -АН, -ЯН, -ИН этот сегмент является суффиксом (исключения: БУЛАНЫЙ, ПОГАНЫЙ, РУМЯНЫЙ, ЕДИНЫЙ, ИНОЙ, СВИНОЙ); среди корней-исключений с написанием буквы О после шипящей удается определить некоторые закономерности (так, О пишется перед следующей гласной У: АНЧОУС, ДЖОУЛЬ, ШОУ), появляется возможность иной формулировки некоторых традиционных правил. Все эти наблюдения можно сделать и не прибегая к моделированию, однако моделирование нацеливает на обнаружение таких частных закономерностей.

В заключение приведем примеры моделей.

Правило: «В суффиксе **-щик(-чик)** имен существительных, обозначающих лицо или предмет, после согласных **д, ж, з, с, т** пишется **ч**, в прочих случаях — **щ**; конечные **к** и **ч** заменяются на **т**: докладчик, перебежчик, смазчик, подписчик, летчик, атомщик, сцепщик, фехтовальщик, кабатчик, передатчик. Исключение: после сочетания **й, н, р, ь** с согласными **д, т** пишется **ш**: шаландщик, алебардщик, флейтщик, процентщик, асфальтщик».

Модель

Обозначения

1. +: д ж з с т
2. —: й н р ь

Правила

1. К*→Т*
2. Ч*→Т*
3. —+*→ Щ
4. +*→ Ч
5. *→ Щ

кабак*ик	передач*ик	процент*ик	лет*ик	гон*ик
кабат*ик	—	—	—	—
—	передат*ик	—	—	—
—	—	процентщик	—	—
кабатчик	передатчик	—	летчик	—
—	—	—	—	гонщик
кабатчик	передатчик	процентщик	летчик	гонщик

Правило: «Для построения формы 1-го л. ед. ч. от инфинитива глаголов с суффиксами **-ов- (-ев-)** после **ц** и шипящих **ж, ч, ш** заменить конечное **-евать** на **-ую**, в прочих случаях заменить **-евать** на **-юю**, **-овать** на **-ую**: танцевать — танцую, бушевать — бушую; горевать — горюю, рисовать — рисую. Исключение: увещевать — увещеваю».

Модель

Обозначение

W : ц ж ч ш

Правила

1. увещевать→увещеваю
2. Wевать#→ую#
3. евать#→юю#
4. овать#→ую#

увещевать	танцевать	бушевать	горевать	рисовать
увещеваю	—	—	—	—
—	танцую	бушую	—	—
—	—	—	горюю	—
—	—	—	—	рисую
увещеваю	танцую	бушую	горюю	рисую

Пример. Последовательные исторические изменения формы инфинитива от праславянского языка к современному русскому. Для упрощения праславянские формы записаны русскими буквами. Горизонтальными штриховыми линиями разделены фрагменты модели, каждый из которых отражает самостоятельное фонетическое (или графическое) изменение.

Модель

Обозначение

J: ж ч ш щ —

Правила

1. ске→ще
2. ке→че
3. ге→же
4. хе→ше

— — — — —

5. Je→a

— — — — —

6. ьр→ер

— — — — —

7. I-а→я
8. ти#→ть#

пискети	крикети	дьргети	дыхети	стоI-ети	видети
пищети	—	—	—	—	—
—	кричети	—	—	—	—
—	—	дьржети	—	—	—
—	—	—	дышети	—	—
пищати	кричати	дьржати	дышати	сто I-ати	—
—	—	держати	—	—	—
—	—	—	—	стояти	—
пищать	кричать	держать	дышать	стоять	видеть

Ю. ЕРМАКОВ

Анализ ответов: метод динамических таблиц

Эффективность компьютерного обучения во многом зависит от формы организации диалога между обучаемыми и персональным компьютером (ПК). В процессе контролирования полученных знаний учащийся должен отвечать на поставленные ПК вопросы, причем формы ответов могут быть различными, начиная от простейших «да», «нет» и кончая предложениями разговорной речи. Метод динамических таблиц позволяет приблизить уровень общения учащегося и ПК к обычному речевому диалогу между учащимся и преподавателем.

Первые простейшие машинные экзаменаторы («Ласточка», «Атом» и др.) были рассчитаны на выбор учащимся в качестве ответа одного из предлагаемых ему вариантов. Такие экзаменаторы имели высокую вероятность случайного правильного ответа, обладали статичностью (код правильного ответа сохранялся). Форма организации взаимодействия между учащимся и машинным экзаменатором была далека от диалога «преподаватель — учащийся». Эти недостатки скомпрометировали применение подобных устройств.

Появление ЭВМ открыло новые перспективы развития автоматизированных обучающих систем, в частности экзаменационных систем (ЭС). В первых системах использовались прежние принципы, но вводилась динамичность (при повторном обращении к одному и тому же вопросу код правильного ответа изменялся). Появились ЭС, где учащийся, отвечая на вопрос, вводит в ЭВМ непосредственно предложение, которое соответствует его речевому ответу в работе с преподавателем. Остро встала проблема распознавания правильности свободно конструируемых текстовых ответов на естественном языке. Для реализации обучающих систем на базе ЭВМ и функционирования в их со-

ставе ЭС разработаны специальные языки программирования: «Автор» [1], «Контакт» [2] и другие. Для анализа и распознавания правильности текстовых ответов использовался, например, метод поиска элементов массива ключевых слов с синонимами, включающий редактирование ответов, процентное сравнение [1, 3], другие методы [2]. Однако ЭС, основанные на непосредственном вводе текста ответа, не показали достаточной эффективности и не получили широкого распространения.

Эффективной ЭС может быть лишь при соблюдении целого ряда требований:

язык диалога должен быть максимально близок к естественному языку общения учащегося и преподавателя;

время ввода ответа не должно превышать времени речевого ответа учащегося на вопрос преподавателя;

оценка правильности ответа должна быть определенной и надежной;

система должна быть динамичной.

Ввод текстового ответа с помощью клавиатуры ПК требует много времени, так как ЭС рассчитаны не на профессиональных машинисток, а на массового пользователя, который обычно не имеет соответствующих навыков. Правда, сейчас разрабатываются системы речевого ввода, но они требуют высокого быстродействия ЭВМ, большого ОЗУ и пока недостаточно надежны. В ближайшем будущем такие системы могут иметь лишь ограниченное применение.

Еще одним недостатком ЭС с текстовым вводом является неоднозначность оценки правильности ответа. Существующие методы не могут гарантировать достоверного и надежного функционирования ЭС: естественные языки настолько многогранны, что практически невозможно учесть все допустимые комбинации слов, которые могут использоваться учащимся; а ведь не исключены и грамматические ошибки.

Вспомним, что ЭС ожидает поступления не произвольного ответа на конкретный вопрос, а вполне определенного, правильного ответа, хотя и здесь возможны варианты. Поэтому будет вполне естественным, если машина предложит учащемуся сконструировать свой ответ из ограниченного набора предлагаемых слов.

Группа слов на экране дисплея представляется в виде таблицы с пронумерованными строками и столбцами. В каждой строке присутствует только одно слово, соответствующее правильному ответу, все остальные слова — «неправильные». Учащийся выбирает из каждой строки по одному слову и вводит в ЭВМ его номер в строке, так что время ввода максимально приближено ко времени речевого ответа на вопрос. Правильность ответа оценивается по совпадению последовательности введенных цифр с эталонной последовательностью и имеет строгую достоверность. Отметим, что предложенная форма ответа не удовлетворяет синтаксическим правилам языка, но соблюдение последних приводит к необходимости дополнительных связей между словами различных строк. Эти связи могут служить одним из источников «подсказок» учащемуся. Полное соответствие синтаксическим правилам заметно усложняет процесс составления таблиц.

При следующем обращении к вопросу положение слов в каждой строке изменяется, но строки остаются в том же порядке, отвечающем порядку слов в ответе.

Будем называть этот метод организации диалога в ЭС методом динамических таблиц.

Рассмотрим пример. На экран терминала выводятся вопрос, таблица слов и запрашивается ответ.

Студент — кто это (в социальном смысле)?

	1	2	3
Студент —			
I	СЛУЖАЩИЙ	УЧАЩИЙСЯ	РАБОЧИЙ
I	ВУЗ	ЗАВОД	КОНТОРА

ВАШ ОТВЕТ?...

В таблице представлены основные ключевые слова ответа, между строками полезно в сомнительных местах вставлять группы слов-связок, облегчающих ориентацию и выбор, но не являющихся «подсказкой» (вспомогательные слова-связки выделяются строчными буквами). В данном случае верный ответ — 2, 1.

Вероятность случайного правильного ответа определяется числом возможных вариантов, которые могут быть составлены с использованием слов таблицы. Так, для таблицы 4×4 число вариантов равно $4^4=256$, поэтому угадывание маловероятно.

На основе метода динамических таблиц в Московском институте радиотехники, электроники и автоматики разработана ЭС для контроля знаний студентов (приема зачетов и лабораторных работ) по курсу «Механика». Программы написаны на Бейсике-3А, функционируют на базе ПК Д3-28 и дисплея 15ИЭ-00-13 (система 15ИПГ-32) в течение двух лет. Опыт разработки и доводки ЭС показал, что количество слов в строке таблицы должно быть ограничено пятью, а общее количество строк — не более шести — восьми. Глаголы следует писать в неопределенной форме, существительные без падежей и т. д. Работа ЭС не требует высокого быстродействия процессора, большого объема оперативной памяти. Так, для используемой нами машины время ожи-

дания следующего вопроса составляет 1—2 с. Каждый студент отвечает на один билет, в который входит четыре вопроса. Используется программная память объемом 20 К, которая достаточна для организации ЭС, имеющей 17 билетов.

Одним из трудоемких и ответственных этапов разработки ЭС является составление таблиц. Именно их содержание определяет качество и эффективность работы ЭС. Заниматься этим должен преподаватель, хорошо знающий курс. Не следует стремиться делать таблицы сложными и громоздкими — в противном случае диалог между ПК и учащимся превращается в «разгадывание ребуса», занимает много времени, эффективность работы и отношение к ЭС резко падает.

Наш опыт применения ЭС свидетельствует о высокой эффективности метода динамических таблиц. Он повышает качество знаний и подготовки студентов, так как ЭС требует твердого знания определений, формулировок теорем и т. д., «Приблизительно» верные ответы не принимаются. Благодаря этому повышается ответственность учащегося при подготовке к работе с ЭС.

Литература

1. *Алексеев В. Е., Булатова И. Г., Савельев А. Я.* Описание языка «Автор» / Математические и информационные проблемы прогнозирования и управления наукой. Киев. Наукова думка, 1971. С. 123—136.

2. *Ницецкий Л. В., Новицкий Л. П.* Представление форм ответов в диалоговой системе обучения и контроля знаний средствами языка «Контакт» // Диалоговые системы. Вып. 2.— Рига: Зинатне, 1978. С. 109—119.

3. *Савельев А. Я.* Автоматизированные обучающие системы на базе ЭВМ / Автоматизированные системы обучения на базе ЭВМ. М.: Знание, 1982. С. 3—33.

О. ПАРСАДАНОВ, Е. КЛЕЦКО

Система тестового контроля знаний учащихся на БК-0010

Настоящая система реализована на ПЭВМ, входящих в состав класса КУВТ-86 (Бейсик). Все программы системы написаны на ПЭВМ БК-0010 на языке Бейсик с использованием элементов программирования в машинных кодах.

Система предназначена для проведения зачетных опросов, а также используется для проведения анкетирования учащихся с последующей статистической обработкой ответов. Несмотря на то что из-за ограниченности ресурсов БК-0010 все вопросы, используемые в тестовых заданиях, относятся к типу так называемых эталонных, т. е. в памяти ЭВМ в той или иной форме содержатся эталоны правильных ответов, гибкое использование такого типа вопросов позволяет получать достаточное (по крайней мере с точки зрения учеников) разнообразие заданий.

При проведении тестовых опросов с учительского места на ЭВМ учеников посылается программа теста, которая содержит блок выбора варианта (от шести до десяти вопросов) и блок передачи результатов на учительское место с одновременной их печатью на принтере. По окончании занятия у преподавателя, таким образом, готов список учеников, прошедших опрос, и их оценки. Эта же информация может быть одновременно записана на ГМД, чтобы затем перенести ее в соответствующий файл системы «Электронный журнал», создание которой сейчас заканчивается.

Блок выбора варианта состоит из короткой подпрограммы, которая под управлением генератора случайных чисел ищет номер варианта и считывает соответствующие значения или выражения для подстановки их в типовые вопросы, а также правильные ответы на эти вопросы из соответствующего файла данных, который хранится на ГМД на ЭВМ учителя.

Как уже было замечено, сами вопросы могут быть весьма разнообразными, хотя с формальной точки зрения все они имеют один тип. В системе использованы четыре основные формы вопросов.

Пример формы 1. Указать номер строки программы, содержащей ошибку.

Приведена программа на языке Бейсик, состоящая 4—5 строк.

Ученик должен набрать номер ошибочной строки. В этом случае требуется ввести однозначный единственный ответ, который, в зависимости от сути вопроса, может быть

числом, словом, именем оператора и т. п. Пример формы 2.

Чему равны переменные A и B после выполнения программы?

```
10 A=4
20 B=B+1
30 A=B+5
40 B=A—B
```

Ученик должен ввести сначала значение A, затем значение B. В этом случае требуется ввести по очереди два однозначных ответа, которые могут быть в зависимости от сути вопроса числом, словом, именем оператора и т. п.

Пример формы 3.

Указать номер верного ответа из предложенных вариантов ответа на вопрос.

Ученик выбирает номер одного из напечатанных на экране вариантов ответа; форма 3 отличается от формы 1 тем, что правильными являются два из 4—5 предложенных вариантов, т. е. ученик должен при ответе проявить еще и способность выбора наиболее оптимального варианта.

Пример формы 4.

Указать имя оператора языка Бейсик, который позволяет ввести данные в память. (Указать имя недостающего в программе оператора.)

В этом конкретном случае ученик может ввести либо имя оператора присваивания LET, либо имя оператора ввода данных с клавиатуры INPUT. Оба ответа будут верными. Форма 4 отличается от формы 3 тем, что на экране нет подсказок в виде вариантов ответов.

После ответа на каждый вопрос ЭВМ повторяет на экране ответ ученика и запрашивает его согласие на обработку этого ответа. Ученик может ответить «нет», и вопрос повторяется снова. Такая процедура необходима, так как часто ученики, не имея устойчивых навыков работы с клавиатурой, могут работать не с тем регистром клавиатуры, могут просто излишне поторопиться и т. д. Словом, неправильность ответа может объясняться не только отсутствием знаний, но и другими причинами, связанными с необычностью метода опроса. Если ученик ответил «да», ответ поступает на подпрограмму анализа ответа, и на экране появляется информация:

```
Задано вопросов     — XX
Верных ответов      — XX
Неверных ответов    — XX
```

Затем появляется очередной вопрос теста.

После ответа на все вопросы фамилия и ответы ученика передаются на учительское место, где происходит печать или запись результатов. После вывода результата на экране ученика и на ЭВМ учителя ученик имеет возможность посмотреть свои собственные ответы на все вопросы, записать их и попробовать определить свои ошибки.

Последний вариант системы полностью исключает возможность посмотреть текст самой программы на ЭВМ ученика, что достигается маленькой подпрограммой блокировки клавиши S в машинных кодах. Таким образом, понимающие суть программы ученики не могут внести в нее изменения, например поставить себе оценку.

Для подготовки программы теста используется специальная программа «формирователь», которая вызывается преподавателем на БК-010. После ее запуска пользователь-преподаватель, отвечая на вопросы программы, задает формулировку и ответ (ответы) каждого вопроса теста. При работе программы «формирователь» вопросы и ответы передаются на ДВК-2, где идет автоматическое создание собственно программы теста.

Таким образом, подготовка нового теста сводится только к ответам на вопросы программы «формирователь». Это позволяет работать с системой не только преподавателям ИВТ, но и учителям-предметникам, не знакомым с программированием.

Систему можно использовать также для проведения анкетирования с последующей статистической обработкой ответов. Для этого меняется блок передачи информации на ЭВМ учителя. Новый блок передачи обеспечивает считывание данных из файла-накопителя ответов на ГМД ЭВМ учителя, анализ ответов и пересылку обратно данных, включающих в себя ответы на конкретной машине ученика. Для обработки результатов используется специальная программа, которая проводит расчет процента соответствующих ответов на каждый вопрос. Сами вопросы обычно имеют вышеописанные формы.

При создании системы большое внимание уделялось вопросам наглядности отображения информации и удобства работы. Активно использовались возможности воспроизведения цветов на БК-0010. Сами вопросы имеют комплексный характер, т. е. для ответа на каждый из них требуется иметь понятие о всем пройденном к моменту опроса материале. Особый интерес, на наш взгляд, представляют вопросы, составленные по формам 3 и 4, которые ставят ученика перед необходимостью выбора единственного ответа из двух правильных.

Система тестового опроса разрабатывалась и применялась в течение 1987/88 учебного года в межшкольном кабинете ИВТ Ленинградского района Калининграда, в котором проходили обучение около 800 учеников десятых классов 12 школ района. С ее по-

мощью были проведены четыре опроса каждого ученика в течение года; кроме того, она использовалась на дополнительных занятиях и для проведения анкетирования.

В результате анкетирования, также проводимого с помощью системы, выяснилось, что около 80 % учеников считают такой способ опроса полезным и более интересным, чем обычные письменные контрольные работы. Следует отметить, что существующая система локальной сети класса КУВТ-86 (Бейсик) весьма несовершенна и медлительна, что не дает возможности полностью использовать преимущества ЭВМ, объединенных в единую сеть.

В настоящее время в кабинете совместно со специалистами Калининградского центра НТТМ «Вектор» заканчивается разработка документации по создаваемому комплексу «Класс», который включает в себя вышеописанную систему «Тест», систему «Электронный журнал» на основе базы данных «Микро» и программные средства, позволяющие напрямую вводить оценки, полученные учениками при автоматическом опросе, в «Электронный журнал».

Со второго полугодия 1988 г. центр НТТМ «Вектор» (236000, г. Калининград, ул. Космонавта Леонова, 24) осуществляет поставку данного ППС всем заинтересованным организациям.

Е. ГЕЛЬТИЩЕВА, Г. СЕЛЕХОВА,
Московский ордена Трудового Красного Знамени научно-исследовательский институт гигиены им. Ф. Ф. Эрисмана

Гигиеническое обоснование сочетания цветов на видеотерминале

Цель описываемых исследований — определение оптимальных с гигиенической точки зрения сочетаний цветов фона и знаков при разработке ППС для ПЭВМ «Ямаха».

Исследования проводились в классе, ориентированном на север. Расположение рабочих мест — периметральное (по двум сторонам учебной аудитории). Работа учащихся осуществлялась при обычном освещении венгерскими потолочными люминесцентными светильниками с экранирующими решетками, вмонтированными в подшивной звукопоглощающий потолок. При включении осветительных установок исключалась отраженная блескость на экранах мониторов. На рабочей поверхности стола создавалась хорошая освещенность порядка 500 лк, на экране — 300 лк, что соответствовало разработанным нами гигиеническим требованиям при работе с видеотерминалами. За каждым учащимся было закреплено определенное рабочее место. До и во время работы учащихся осуществлялась широкая аэрация класса через окна. В период работы они были зашторены. Температура и относительная влажность воздуха в период исследований не изменялись и находились в пределах допустимых величин (температура 18—22°С, относительная влажность выше 31 %), согласно «Методическим указаниям по профилактике переутомления студентов вузов при работе с видеотерминалами», утвержденным Минздравом СССР 05.03. 1988 г. № 4563 и Минвузом СССР 09.03 1988 г. № 96-21-05/65.

Под наблюдением находились 8 учащихся, окончивших IX класс. По каждому из изучаемых показателей проведено 100 исследований.

Изучение состояния здоровья испытуемых по данным углубленного медицинского обследования позволило отнести их к I и II группам здоровья. Все имели нормальное цветоощущение.

Время работы каждого учащегося за видеотерминалом составляло 15 мин. Оно определялось развитием адаптации зрительного анализатора и временем, не превышающим регламентируемые величины при работе на видеотерминалах III класса, согласно рекомендациям «Режим занятий учащихся за видеотерминалами», утвержденным Минздравом СССР и Минпросом СССР 05.02.1987 г. № 29 - 38/14.

Физиолого-гигиенические исследования осуществлялись в утренние часы (от 10 до 11 ч). Каждый учащийся обследовался один раз в день. Во время работы учащихся за видеотерминалом уделялось внимание соб-

людению правильной посадки и сохранению оптимального расстояния от глаз до экрана (60—70 см).

Состояние учащихся при работе на видеотерминалах с различной цветовой индикацией экрана и знаков оценивалась как по субъективным, так и по объективным показателям функционального состояния зрительного анализатора. Ребятам предлагалось дать оценку сочетания цветов экрана и знаков («хорошее», «удовлетворительное», «плохое», «нравится», «не нравится»). К субъективным показателям относились и оценки общего самочувствия («немного устал», «не устал») и состояния зрительного анализатора (глаза «не устали», «устали средне», «устали немного»). Функциональное состояние зрительного анализатора исследовалось методом определения критической частоты слияния световых мельканий (КЧССМ) и аккомодационной способности глаза с последующим вычислением коэффициента утомляемости по С. Л. Шаповалову. Опрос учащихся и исследование функции зрительного анализатора проводились до и после работы на ПЭВМ «Ямаха».

В начале исследования предлагались следующие сочетания цветов экрана и знаков: темно-зеленый фон и белые знаки, темно-зеленый фон и светло-зеленые знаки, светло-зеленый фон и красные знаки, синий фон и белые знаки, синий фон и светло-зеленые знаки, синий фон и красные знаки, белый фон и черные знаки, черный фон и белые знаки, черный фон и зеленые знаки, черный фон и розовые знаки.

В подборе отдельных сочетаний принимали участие учащиеся. Однако, несмотря на это, некоторые наблюдаемые отказывались работать при сочетаниях: светло-зеленый фон и красные знаки, синий фон и светло-зеленые знаки, синий фон и красные знаки, черный фон и розовые знаки. Отказ от работы даже одного наблюдаемого не позволял нам считать благоприятными соответствующие сочетания цветов, поэтому в дальнейшем они были исключены. Все учащиеся отказывались работать за видеотерминалом при белом фоне и черных знаках, хотя в литературе отмечается, что такое сочетание с эргономических позиций является благоприятным. При этом они заявляли, что «не хотят портить глаза». С большим трудом удалось уговорить учащихся работать при положительном контрасте изображения. Как показали наши наблюдения, негативную реакцию можно объяснить заметным мерцанием экрана, которое менее выражено при более темном фоне экрана.

Таким образом, для исследования были отобраны 6 сочетаний: темно-зеленый фон и белые знаки, темно-зеленый фон и светло-зеленые знаки, черный фон и белые знаки, черный фон и зеленые знаки, синий фон и белые знаки, белый фон и черные знаки.

При анализе данных мы исходили из того, что субъективная оценка достаточно объективно и полно может характеризовать предъявляемые сочетания цветов экрана и знаков, общее самочувствие и состояние зрительного анализатора.

В табл. 1 приводятся субъективные оценки учащихся для различных сочетаний цветов экрана и знаков, выраженные в процентах. Только одно сочетание — темно-зеленый фон и белые знаки — отмечено всеми учащимися как «хорошее», с которым им работать «нравится».

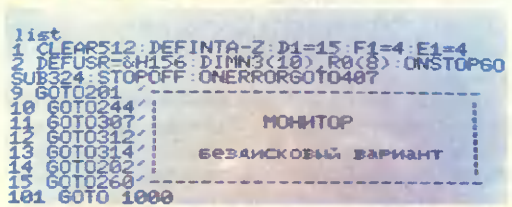
Наихудшим сочетанием учащиеся считают белый фон и черные знаки, что, как указывалось выше, обусловлено мерцанием экрана, поскольку частота кадровой развертки составляет 50 Гц.

Субъективные оценки общего самочувствия и состояния зрительного анализатора представлены в процентах в табл. 2.

Здесь, так же как и в табл. 1, наилучшие показатели по состоянию зрительного анализатора и общему самочувствию получены при сочетании темно-зеленого фона и белых

Таблица 1

№ п/п	Сочетания цветов экрана и знаков	Оценка, %				
		хорошее	удовлет.	плохое	нравится	не нравится
1	Темно-зеленый фон и белые знаки	100	—	—	100	—
2	Темно-зеленый фон и светло-зеленые знаки	20	20	60	33	67
3	Черный фон и белые знаки	14	72	14	29	71
4	Черный фон и зеленые знаки	25	75	—	83	17
5	Синий фон и белые знаки	50	50	—	50	50
6	Белый фон и черные знаки	—	29	71	—	100



знаков, наихудшие по состоянию зрительного анализатора — белый фон и черные знаки, по общему самочувствию — только при синем фоне экрана и белых знаках.

Результаты обследования функционального состояния зрительного анализатора подвергались обработке при помощи однофакторного дисперсионного анализа. Обработка результатов осуществлялась на ЭВМ. После проведения дисперсионного анализа устанавливались 95 %-ные доверительные интервалы для средних значений: $\bar{X} \pm t_{0.05} S_{\bar{x}}$, где $\bar{X}$ — среднее по экранам, $S_{\bar{x}}$ — нормированная ошибка среднего, $t = 1{,}66$ — критерий Стьюдента для 5 %-ного уровня значимости (см. табл. 3).

Результаты дисперсионного анализа показали существенно значимые различия средних величин предъявляемых цветовых сочетаний.

КЧССМ — временной показатель возбудимости зрительного анализатора. Его снижение свидетельствует о замедлении во времени каждого цикла возбуждения, т. е. об ухудшении функционального состояния. Некоторое улучшение КЧССМ при работе на экране с темно-зеленым фоном и светло-зелеными знаками несущественно и находится в пределах ошибки измерения. По всей вероятности, это можно объяснить повышенной чувствительностью зрительного анализатора к зеленому цвету.

При анализе сдвигов по коэффициенту утомляемости наблюдается разнонаправленность сдвигов при сочетании цвета экрана с белым фоном и черными знаками и ухудшение (увеличение) коэффициента утомляемости при работе учащихся с темно-зеленым фоном и светло-зелеными знаками. Такое увеличение коэффициента утомляемости мы склонны объяснить недостаточной контрастностью информации на экране.

Полученные нами сравнительно небольшие изменения функционального состояния зрительного анализатора учащихся при работах за видеотерминалами объясняются, во-первых, тем, что во время исследования у учащихся не было учебных занятий и им не надо было готовить домашних заданий; во-вторых, положительной мотивацией (для занятий отбирались ребята, которые хотели составлять учебные программы) и, в-третьих, минимальным временем работы на ПЭВМ «Ямаха».

По результатам, представленным в табл. 1, 2, 3, были составлены ряды предпочтительности, в которых сочетания цветовой индикации экрана располагались в порядке возрастания субъективных оценок учащихся (ряды с 1—4) и в порядке улучшения функционального состояния (ряды с 5—6). При построении рядов предпочтительности цифрами обозначены порядковые номера сочетаний цвета экрана и знаков в соответствии с табл. 1, 2, 3.

Ряд предпочтительности по табл. 1:

по оценке «хорошее» сочетание цветов экрана и знаков $1>5>4>2>3>6$ (1);

по выбору сочетаний «нравится» $1>4>5>2>3>6$ (2).

Таблица 2

№ п/п	Сочетания цветов экрана и знаков	Оценка, %				
		Состояние зрительного анализатора			Общее самочувствие	
		Глаза «не устали»	Глаза устали		«Немного устал»	«Не устал»
			«средне»	«немного»		
1	Темно-зеленый фон и белые знаки	100	—	—	—	100
2	Темно-зеленый фон и светло-зеленые знаки	40	—	60	—	100
3	Черный фон и белые знаки	57	14	29	—	100
4	Черный фон и зеленые знаки	50	—	50	—	100
5	Синий фон и белые знаки	32	25	43	13	87
6	Белый фон и черные знаки	14	43	43	—	100

Таблица 3

Изменение показателей функционального состояния зрительного анализатора

№ п/п	Сочетания цветов экрана и знаков	КЧССМ, Гц	Коэффициент утомляемости, %
		$\bar{X} \pm t_{0.05} S_{\bar{x}}$	$\bar{X} \pm t_{0.05} S_{\bar{x}}$
1	Темно-зеленый фон и белые знаки	$-1,0 \pm 0,32$	$-1,5 \pm 3,2$
2	Темно-зеленый фон и светло-зеленые знаки	$+0,3 \pm 0,71$	$+3,0 \pm 6,8$
3	Черный фон и белые знаки	$-1,7 \pm 0,65$	$-7,3 \pm 6,3$
4	Черный фон и зеленые знаки	$-1,7 \pm 0,65$	$-7,4 \pm 5,9$
5	Синий фон и белые знаки	$-3,9 \pm 0,61$	$-4,5 \pm 5,9$
6	Белый фон и черные знаки	$-3,0 \pm 0,65$	$+0,1 \pm 6,3$

Примечание. знак «+» указывает на увеличение сдвига, знак «—» на снижение.

Ряд предпочтительности по табл. 2:
по состоянию зрения «глаза не устали» 1>3>4>2>5>6 (3);
по общему состоянию «не устал» 1, 2, 3, 4 и 6>5 (4).

Ряд предпочтительности по табл. 3:
по КЧССМ — 1 и 2>3 и 4>5 и 6 (5);
по коэффициенту утомляемости — 1 и 6>>5>3 и 4>2 (6).

Каждому из членов ряда предпочтительности присвоено определенное место, лучшему — 1-е и т. д. Согласно распределенным местам рассчитывается средняя оценка предпочтительности (см. табл. 4).

Как видно из табл. 4 на первом месте по предпочтительности с учетом состояния зрительного анализатора и общего самочувствия после работы на видеотерминале стоит сочетание темно-зеленый фон и белые знаки, на втором месте — черный фон и зеленые

Таблица 4

Средняя оценка предпочтительности по субъективным и объективным показателям состояния зрительного анализатора и общего самочувствия

№ п/п	Сочетания цветов экрана и знаков	Средняя оценка предпочтительности по порядку мест					
		1	2	3	4	5	6
1	Темно-зеленый фон и белые знаки	1,0	—	—	—	—	—
2	Темно-зеленый фон и светло-зеленые знаки	—	—	—	3,0	—	—
3	Черный фон и белые знаки	—	—	2,8	—	—	—
4	Черный фон и зеленые знаки	—	2,3	—	—	—	—
5	Синий фон и белые знаки	—	—	2,8	—	—	—
6	Белый фон и черные знаки	—	—	—	—	3,8	—

знаки, на последнем месте — белый фон и черные знаки.

Если исключить из рядов предпочтительности показатель общего самочувствия, а основываться только на субъективных и объективных показателях, характеризующих функциональное состояние зрительного анализатора, то последовательность мест по порядку предпочтительности в табл. 1, 2, 3 будет следующая с учетом порядковых номеров сочетания цветов 1>4>2 и 5>3>6. Это особенно важно учитывать, так как при работе на видеотерминалах большая нагрузка приходится на зрительный анализатор. Необходимо выбирать такое сочетание цвета экрана и знаков, которое вызывает его наименьшее утомление.

Таким образом, на основании проведенного исследования можно сделать вывод, что наилучшим сочетанием с гигиенической точки зрения при работе учащихся за видеотерминалами ПЭВМ «Ямаха» является **темно-зеленый фон экрана и белые знаки.**

А. БАТАРШЕВ, А. РООСИ

Учебный автоматизированный измерительный комплекс

В профессиональной школе менее всего разработан вопрос об обучающих программах, изменяющих способы предъявления информации в зависимости от способностей и уровня знаний обучаемых.

В Таллинском политехническом институте разработана имитационная модель автоматизированного измерительного комплекса (АИК), который может быть адаптирован к обучаемым в процессе выполнения ими лабораторных работ. АИК базируется на персональной микро-ЭВМ «Искра-226» и содержит в своем составе промышленные измерительные приборы и источники сигналов, полностью управляемые ЭВМ по каналу общего пользования (КОП), специально разработанному для малых систем, работающих под управлением микро-ЭВМ. Эти системы широко используются в промышленности, в научных исследованиях (АСНИ) и в сфере обслуживания. Для создания разнообразных структурных связей в комплексе используется многоканальный матричный коммутатор аналоговых сигналов, управляемый по КОП. Это позволяет проводить лабораторные работы по измерению параметров различных электрических схем. Для осуществления диалога с пользователем системы (оператором) используется символьно-графический дисплей с клавиатурой и матричное графическое печатающее устройство (матричный принтер). Имеется возможность подключения датчика ввода графической информации. Создано развитое программное обеспечение для управления измерениями в ручном и автоматическом режимах, для сбора, обработки и вывода результатов измерений, а также для диалогового взаимодействия оператора с комплексом.

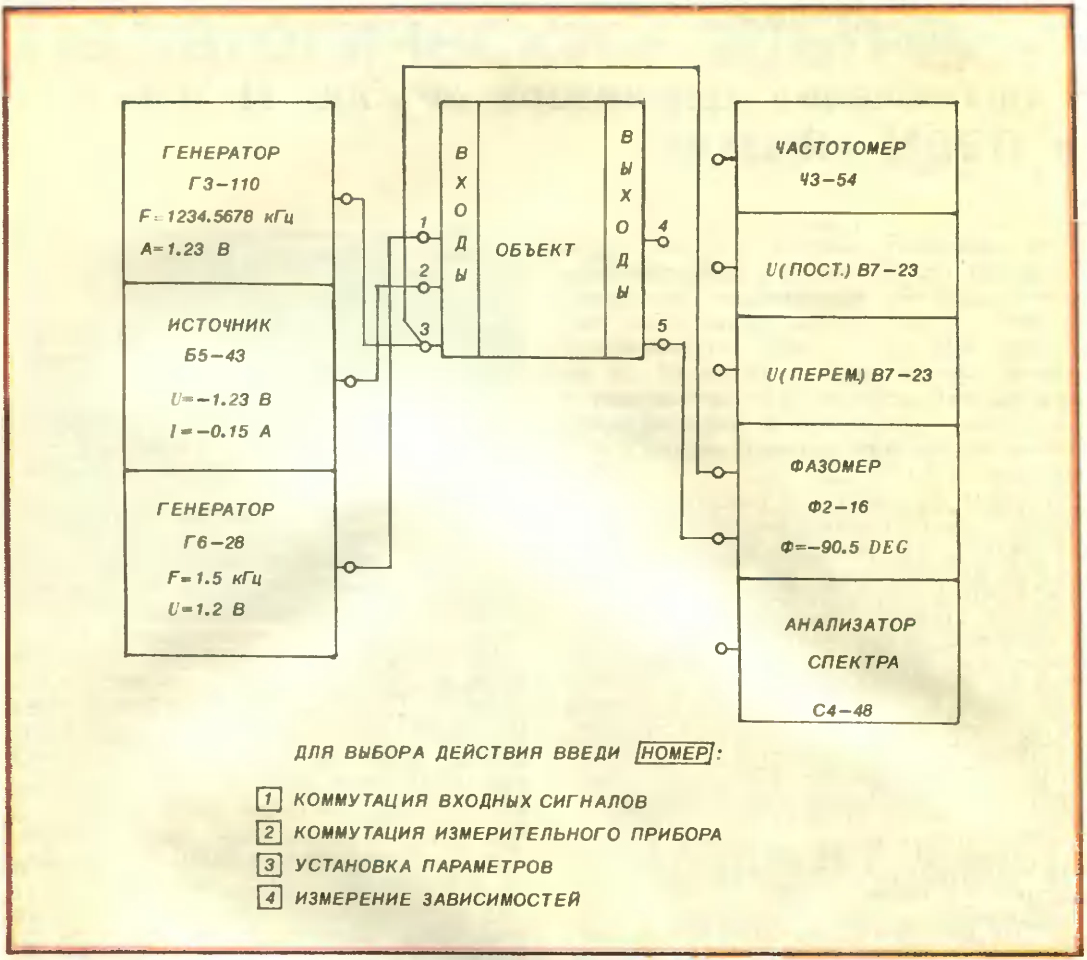
В результате у операторов формируются первоначальные навыки по вводу и чтению информации, по управлению системой в различных режимах ее работы. Вначале на экране дисплея высвечиваются указания по управлению режимами работы системы, учащимся предлагается приступить к вводу данных и управляющих команд. Операторы знакомятся с общей структурой программы. При правильной работе оператора дополнительные сведения на экране не сообщаются, однако всю необходимую информацию он может получить, вводя на произвольном этапе выполнения измерений или вычислений специальные команды. Ошибочный ввод данных сопровождается звуковым сигналом и появлением на экране вопросительного знака, а также разъяснения причины ошибки пользователя. При необходимости на экране высвечивается «подсказка» о допустимых пределах вводимых параметров.

Пользователями АИК являются студенты первого курса института, а в перспективе ими будут учащиеся профтехучилищ.

В процессе эксплуатации АИК выявлены следующие особенности.

Так как первоначальные тренировки по управлению АИК проводятся неподготовленными пользователями, основным режимом работы системы на этом этапе является режим подсказки.

Для непосредственного выполнения задания обучаемым дополнительно требуется информация о состоянии функциональной схемы АИК, о параметрах входных сигналов, о рабочих режимах измерительных приборов. Представление этих сведений на экране дисплея требует дополнительной поясняющей информации, которая выводится в виде функ-



циональной схемы. Используя эту информацию, пользователи проводят измерения требуемых параметров. Пример изображения одной из функциональных схем показан на рисунке. На схеме слева отображаются сигнальные параметры, справа — результаты измерений. Вводя соответствующие управляющие команды (высвечиваются на экране внизу), оператор может изменить отображаемые параметры на функциональной схеме.

Восприятие результатов измерений будет более эффективным при предъявлении их в графическом виде. В ряде случаев оператора интересуют числовые значения некоторых точек. Для этого одновременно с графическим представлением результатов измерений в АИК используется указатель графической информации. Числовые значения интересуемых величин в этом случае отображаются в нижней части экрана. С программами, включающими элементы графического представления данных, обучаемые работают гораздо успешнее.

В соответствии с данными предварительного эксперимента время выполнения обучаемыми лабораторных работ в экспериментальных группах по сравнению с контрольными сократилось в среднем на 20 %, а количество допущенных ошибок — на 30 %.

Литература

1. *Батаршев А. В.* К вопросу о факторах, влияющих на надежность операторов автоматических и полуавтоматических линий станков и установок поточного производства // Эффективность, качество и надежность эрготехнических систем. М.: АН СССР, 1981. С. 246.

2. *Милерян Е. А.* Психологический паспорт машины // Очерки психологии труда оператора. М.: Наука, 1974. С. 290—307.

3. *Хейнрихсен В. Р., Рооси А. Р.* Автоматизированный измерительный комплекс на базе ЭВМ // Методы и средства цифровой обработки данных. Радиотехника X: Тр. Таллинск. политехн. ин-та. Таллин, 1984. С. 49—54.

Г. ХАНОВА, Л. ВЕРБИЦКАЯ,
НИИ ИВТ АПН СССР

Подключение телевизора «Рубин Ц-208» к ПЭВМ «Ямаха»

При проведении занятий в классе часто возникает необходимость в демонстрационном устройстве, позволяющем показывать графику и текст одновременно всем учащимся. Для этого могут использоваться цветные телевизоры с экраном 61 см по диагонали. К сожалению, кодировки цветов сигналами в компьютере и телевизоре различны. Эту несогласованность можно устранить двумя способами.

Первый заключается в том, чтобы с помощью дополнительного устройства добиться полного соответствия выходных параметров RGB-видеосигналов с входными параметрами видеоусилителей телевизора.

Второй способ — подключение компьютера к телевизору только при амплитудных согласованиях сигналов, без внесения изменений в схему телевизора. В данном случае несоответствие цветности устраняется программно. Сохраняется возможность использовать телевизор для приема телепередач, поэтому такой способ предпочтительнее.

Подключение телевизора по второму способу выполняется согласно таблице соединений.

Точки присоединения обозначены в соответствии с принципиальной электрической схемой телевизора.

Сигнал компьютера «синхросмесь» нуждается в усилении и изменении полярности. Для этого использован усилитель на видеовходе телевизора; предварительно нужно убрать связи его с другими блоками, оставив связь с блоком разверток через разъем X1(A1).

Вывод компьютера	Точка подсоединения в телевизоре
B	Контакт 1 разъема X1 (плата A9) блока выходного видеоусилителя
G	Контакт 1 разъема X1 (A10) блока выходного видеоусилителя
R	Контакт 1 разъема X1 (A11) блока выходного видеоусилителя
Синхросмесь	Контакт 2 разъема X26 (видеовход)
Audio	Контакт 6 разъема X1 (A2) блока УПЧ звука
Земля	Контакт 1 разъема X26 (видеовход)

Сигнальные цепи компьютера подключаются к телевизору через гальванические развязки (конденсаторы емкостью порядка 0,1 мкФ). Все другие проводники, идущие в телевизоре от точек подключения указанных сигнальных цепей на предыдущие схемы, желательно разъединить, что можно реализовать через переключатель. Для восстановления черного уровня нужно верхние (по схеме) выводы переменных резисторов R21, R22 и R23 соединить через сопротивления 4,3 кОм с источником +12 В. При таком подключении телевизора, как уже говорилось, не совпадают спектральные соотношения цветности. Это искажение легко преодолевается коррекцией цветозадающей программы PALITRA и программы PAC.

Простота описанного подключения позволит реализовать его широкому кругу пользователей классов «Ямаха».

Уважаемые читатели!

У центра НТТМ «Мёте», поставляющего систему «РИГА-микро» для КУВТ-86 (см.: Информатика и образование. 1988. № 6), изменился телефон. Новый номер — 26-510.

В. ГОРЛЕНКО,
МГПИ им. В. И. Ленина

ЭВМ и дидактические игры

В настоящее время компьютеризация обучения — генеральное направление перестройки высшей и средней школы в нашей стране. На начальном этапе выдвигаются задачи овладения компьютерной грамотностью и формирования положительного отношения учащихся к ВТ. Решение этих, на первый взгляд несложных задач требует учета целого ряда психолого-педагогических факторов, так как воздействие ЭВМ на личность человека, в том числе на его эмоциональную, мыслительную и т. п. деятельность, может быть непредсказуемым (возникновение страха перед машиной, подавление интуитивного начала в мышлении и т. п.) [1]. Одним из способов, формирующих положительную мотивацию у обучаемого к ЭВМ и помогающих преодолеть психологический барьер, является использование *компьютерных игр.*

В настоящее время большинство компьютерных игр не несут смысловой нагрузки и в основном способствуют совершенствованию техники общения человека с машиной на уровне ввода-вывода, хотя игры, построенные на определенном учебном материале, совершенно очевидно должны преследовать педагогические цели. Такие игры можно назвать *компьютерными дидактическими играми.* Обычные учебные компьютерные программы, применяемые в рамках традиционной системы обучения (контролеры, тренажеры и т. п.), в известной степени помогают преодолеть один из ее основных недостатков, а именно пассивное восприятие нового материала. Однако при этом сохраняется другой существенный недостаток традиционной системы — слабая мотивация. Компьютерная дидактическая игра, на наш взгляд, является эффективным средством формирования положительной мотивации. В первую очередь она вызывает у учащегося появление внешних, социальных, мотивов (стремление выиграть в соревновании с машиной, престиж, одобрение сверстников и т. п.). Наблюдения тем не менее показывают, что в ходе игры внешние мотивы могут перерастать во внутренние, познавательные (стремление проверить и уточнить свои знания, установить закономерности, вникнуть в суть вопроса и т. д.). Следствием является более глубокое усвоение знаний.

В данной работе приводится пример компьютерной дидактической игры, а именно ЭВМ «угадывает» задуманный играющим объект (из приведенных в списке) при условии, что играющий ответит на ряд вопросов. Вопросы касаются характеристики объекта. При отрицательном результате (ЭВМ «не угадала») машина «указывает» вопросы, на которые были даны неправильные ответы. Идея этой компьютерной игры принадлежит доценту кафедры математической физики МГПИ им. В. И. Ленина Е. К. Годуновой, разработавшей соответствующую программу на языках программирования Фокал и Бейсик [2]. Данная компьютерная игра достаточно универсальна. На ее основе могут быть созданы дидактические игры для различных специальных предметов, от математики и физики до гуманитарных наук.

С небольшими модификациями автор использовал эту программу для разработки дидактической игры по химии, посвященной узловому вопросу — периодической системе Д. И. Менделеева. В этой игре ЭВМ «угадывает» задуманный элемент таблицы. В вопросах, которые «предлагает» ЭВМ для характеристики элемента, делается акцент на установление взаимосвязи между строением элемента, его свойствами и положением в периодической таблице.

Управляющая часть программы после ответа на очередной вопрос разделяет элементы на две группы (не обязательно равные). Серия ответов типа «да—нет» завершается выводом на экран дисплея результата (названия задуманного элемента). Если ЭВМ «не угадала», играющий вводит в машину название задуманного элемента и машина «указывает» вопрос, на который был дан неверный ответ. В качестве иллюстрации приведен ряд кадров с текстом, выводимым на экран. При необходимости в программу можно ввести дополнительную информацию.

1-й кадр

ЗАДУМАЙТЕ ОДИН ИЗ ПЕРВЫХ 56 ЭЛЕМЕНТОВ ПЕРИОДИЧЕСКОЙ СИСТЕМЫ ХИМИЧЕСКИХ ЭЛЕМЕНТОВ Д. И. МЕНДЕЛЕЕВА
ЭВМ ПОПЫТАЕТСЯ ОТГАДАТЬ ЗАДУМАННЫЙ ЭЛЕМЕНТ, ЕСЛИ ВЫ ОТВЕТИТЕ НА НЕСКОЛЬКО ВОПРОСОВ
ФОРМА ОТВЕТА: ДА — НАЖМИТЕ 1 ВК
НЕТ — НАЖМИТЕ 0 ВК
КОГДА БУДЕТЕ ГОТОВЫ — НАЖМИТЕ КЛАВИШУ ВК

2-й кадр

ЭВМ ПОЛАГАЕТ, ЧТО ЗАДУМАННЫЙ ЭЛЕМЕНТ НАХОДИТСЯ В МАЛОМ ПЕРИОДЕ

3-й кадр
(если ответ «да»)

ВНЕШНИМ ЭЛЕКТРОННЫМ УРОВНЕМ ЯВЛЯЕТСЯ 1- ИЛИ 2S-ПОДУРОВЕНЬ?

3-й кадр
(если ответ «нет»)

ВОЗМОЖНО, ЧТО ЧИСЛО ЭЛЕКТРОННЫХ УРОВНЕЙ В ЭЛЕКТРОННОЙ ОБОЛОЧКЕ АТОМА МЕНЬШЕ ПЯТИ?

4-й кадр
(если ответ «да»)

ЭВМ КАЖЕТСЯ, ЧТО ПРОСТОЕ ВЕЩЕСТВО, ОБРАЗОВАННОЕ ЭЛЕМЕНТОМ,— ЭТО ГАЗ ПРИ НОРМАЛЬНЫХ УСЛОВИЯХ (ДА ИЛИ НЕТ?)

5-й кадр
(если ответ «да»)

ЭВМ ПОЛАГАЕТ, ЧТО ЭЛЕМЕНТ ИМЕЕТ НЕЗАПОЛНЕННУЮ S-ОРБИТАЛЬ НА ВНЕШНЕМ УРОВНЕ

6-й кадр
(если ответ «да»)

ЭЛЕМЕНТ, КОТОРЫЙ ВЫ ЗАДУМАЛИ,— ЭТО ВОДОРОД. ЗАДУМАННЫЙ ВАМИ ЭЛЕМЕНТ УГАДАН ПРАВИЛЬНО? (ДА ИЛИ НЕТ?)

7-й кадр
(если ответ «да»)

ВЫ ХОРОШО ЗНАЕТЕ ЭТОТ ЭЛЕМЕНТ И СОВЕРШЕННО ПРАВИЛЬНО ОТВЕТИЛИ НА ВОПРОСЫ. МОЛОДЕЦ! ХОТИТЕ СЫГРАТЬ ЕЩЕ РАЗ?

7-й кадр
(если ответ «нет»)

ВЫ ДОПУСТИЛИ НЕТОЧНОСТЬ В ОТВЕТАХ! НАПЕЧАТАЙТЕ НАЗВАНИЕ ЗАДУМАННОГО ВАМИ ЭЛЕМЕНТА. ЭВМ УКАЖЕТ ВОПРОС, ОТВЕТ НА КОТОРЫЙ БЫЛ НЕТОЧЕН

Описанная компьютерная игра была опробована на занятиях со студентами химических и биологических специальностей пединститута. Программу целесообразно использовать и при изучении химии в школе в старших классах. Работа с программой позволяет отработать навыки обращения с таблицей Д. И. Менделеева и умение охарактеризовать элемент по положению в таблице, уточнить знание периодического закона (в частности, уяснить различия между малыми и большими периодами, соответствие между номером периода и числом электронных оболочек, номером группы и числом внешних электронов, закономерности распределения электронов по подуровням, особенности заполнения электронных оболочек элементов главных и побочных подгрупп, уточнить знание степеней окисления элементов в соединениях, некоторых их свойств и т. п.).

В процессе наблюдения за играющими хорошо прослеживается смена внешней мотивации (стремления выиграть) на внутреннюю, познавательную. Эффект от неожиданно быстрого «угадывания» машиной задуманного элемента сменяется удивлением в связи со «способностью машины» при отрицательном результате проанализировать ответы и указать вопрос, на который был дан неправильный ответ. Появление этого вопроса вызывает естественную реакцию играющего: «Почему ответ неверен?» — и побуждает к выяснению причины ошибки. В поисках правильного ответа возникает желание повторить игру и стремление проверить себя на других примерах. Это свидетельствует уже о внутреннем побуждении к освоению, уточнению и углублению знаний.

Желающие ближе познакомиться с программой могут обратиться по адресу: 119243, Москва, МГПИ им. В. И. Ленина, биолого-химический факультет, ул. Кибальчича, д. 6, корп. 5, Горленко В. А.

Литература

1. *Тихомиров О. К.* Основные психолого-педагогические проблемы компьютеризации обучения // Вопросы психологии. 1986. № 5. С. 67—69.

2. *Годунова Е. К.* и др. Использование ЭВМ в преподавании математического анализа. Тема: ряды: Методические разработки. М.: Изд-во МГПИ им. В. И. Ленина. 1986.

Р. ЕРОХИНА,
Борисоглебский педагогический институт

Е. СТЕПАНОВА,
МГПИ им. В. И. Ленина

ПМК на уроках астрономии

Применение ПМК на уроках астрономии может стать действенным средством интенсификации самостоятельной работы учащихся. ПМК позволяют быстро провести необходимые расчеты и освободить время для обсуждения и осмысления полученных результатов. Особенно эффективно их применение при выполнении заданий, придающих деятельности учащихся характер исследовательской работы и раскрывающих, хотя и упрощенно, некоторые методы научного исследования в астрономии.

Ниже приводятся задания такого типа и методика их выполнения. Программы расчетов написаны для ПМК «Электроника Б3-34».

Задание 1. Построить орбиту малой планеты по ее координатам, рассчитанным с интервалом в 5 суток, если в перигелии она находится на расстоянии 0,5 а. е. от Солнца и имеет скорость 0,026 а. е./сут. Считать, что планета движется под действием притяжения только со стороны Солнца. Влияние других планет не учитывать.

Решение. Основная цель — определить координаты планеты в любой момент времени. Выберем систему координат (рис. 1), начало которой совпадает с Солнцем, а ось X направлена к перигелию планеты. Ось Y должна быть расположена так, чтобы векторы скорости планеты и силы тяготения лежали в плоскости X0Y, тогда координата z планеты всегда будет равна 0, и задача сведется к нахождения координат x и y.

Определим силу, действующую на планету со стороны Солнца:

$$F = G\frac{M_0 m}{r^2},$$

где M_0 — масса Солнца, m — масса планеты, $r=\sqrt{x^2+y^2}$ — радиус-вектор планеты.

Проекции силы F на координатные оси:

$$F_x = -G\frac{M_0 m}{r^2}\cos\alpha,$$

$$F_y = -G\frac{M_0 m}{r^2}\sin\alpha.$$

Так как $\cos\alpha = x/r$, $\sin\alpha = y/r$, то

$$F_x = -G\frac{M_0 m}{r^3}x,$$

$$F_y = -G\frac{M_0 m}{r^3}y.$$

Проекции ускорения планеты на координатные оси:

$$a_x = -G\frac{M_0}{r^3}x,$$
$$a_y = -G\frac{M_0}{r^3}y. \qquad (1)$$

Следует подчеркнуть, что координаты планеты непрерывно меняются, следовательно, меняются и сила тяготения, а также скорость и ускорение планеты. Для решения подобной задачи в общем виде необходимо использовать специальные математические методы, однако приближенно вычислить несколько последовательных положений небесного тела на орбите довольно просто.

Пусть в начальный момент времени t_0 планета находилась в точке (x_0, y_0); ее расстояние от Солнца $r_0=\sqrt{x_0^2+y_0^2}$; проекции скорости и ускорения на координатные оси — v_{0x}, v_{0y}, a_{0x}, a_{0y}. Определим новые координаты планеты (x_1, y_1) в момент $t_1=t_0+\Delta t$, где Δt — достаточно малый промежуток времени, в течение которого силу, действующую на планету, можно считать неизменной, а движение равноускоренным.

Проекции скорости планеты в середине интервала времени Δt, т. е. (при равноускоренном движении) проекции средней скорости на этом интервале, равны

$$v_{1x} = v_{0x} + a_{0x}\Delta t/2,$$
$$v_{1y} = v_{0y} + a_{0y}\Delta t/2.$$

Координаты в момент t_1:

$$x_1 = x_0 + v_{1x}\Delta t,$$
$$y_1 = y_0 + v_{1y}\Delta t.$$

Расстояние от Солнца в момент t_1 равно $r_1=\sqrt{x_1^2+y_1^2}$, а проекции ускорения (в соответствии с (1)) равны

$$a_{1x} = -G\frac{M_0}{r_1^3}x_1,$$

$$a_{1y} = -G\frac{M_0}{r_1^3}y_1.$$

Повторив расчет, можно найти проекции скорости планеты и ее координаты для момента $t_2=t_1+\Delta t$ и т. д. Общие формулы

для момента $t_i=t_{i-1}+\Delta t=t_0+i\Delta t$ таковы:

$$v_{i,x}=v_{i-1,x}+a_{i-1,x}\Delta t/2 \quad (2)$$
$$v_{i,y}=v_{i-1,y}+a_{i-1,y}\Delta t/2 \quad (3)$$
$$x_i=x_{i-1}+v_{i,x}\Delta t \quad (4)$$
$$y_i=y_{i-1}+v_{i,y}\Delta t \quad (5)$$

Используя их, можно вычислить координаты планеты и построить ее орбиту (с учетом сделанных допущений).

Расчеты выполняются по следующей программе.

Адрес	Нажимаемые клавиши	Код
00	ИП А	6—
01	FX^2	22
02	ИП В	6L
03	FX^2	22
04	+	10
05	↑	0E
06	$F\sqrt{}$	21
07	X	12
08	F 1/X	23
09	ИП 1	61
10	X	12
11	П 4	44
12	ИП А	6—
13	X	12
14	ИП 2	62
15	X	12
16	ИП С	6[
17	+	10
18	П С	4[
19	ИП 4	64
20	ИП В	6L
21	X	12
22	ИП 2	62
23	X	12
24	ИП Д	6Г
25	+	10
26	П Д	4Г
27	ИП 5	65
28	П 2	42
29	ИП С	6[
30	X	12
31	ИП А	6—
32	+	10
33	ПА	4—
34	ИП 2	62
35	ИП Д	6Г
36	X	12
37	ИП В	6L
38	+	10
39	П В	4L
40	С/П	50
41	БП	51
42	00	00

Время измеряется в сутках, расстояние — в а. е., за единицу массы принята масса Солнца. При таком выборе единиц измерения числовые значения исходных данных таковы:

$$x_0=0{,}5;\ y_0=0;\ v_{0x}=0;\ v_{iy}=0{,}026;$$
$$-GM_0=-1/58^2;\ \Delta t/2=2{,}5.$$

Перед началом вычислений исходные данные заносятся в следующие регистры памяти:

x_0 в А, y_0 — В, $v_{0,x}$ — С, $v_{0,y}$ — Д,
$(-GM_0)$ — 1, $\Delta t/2$ — 2, Δt — 5.

Включив ПМК, учащиеся в режиме F ПРГ вводят текст программы, затем в режиме F АВТ записывают в память исходные данные и нажимают В/О и С/П. По окончании счета на индикаторе высвечивается значение координаты y; значение x находится в регистре А и вызывается нажатием клавиш ИП А. Для нахождения следующей пары (x, y) нужно нажать С/П и т. д. Результаты вычислений заносятся в таблицу (табл. 1).

Таблица 1

t, сут.	x, а. е.	y, а. е.
0	0,500	0,000
5	0,485	0,130
10	0,442	0,252
15	0,374	0,360
20	0,285	0,450
25	0,183	0,516
30	0,073	0,560
35	—0,040	0,580
40	—0,152	0,579
45	—0,259	0,557
50	—0,357	0,518
55	—0,445	0,463
60	—0,520	0,395
65	—0,582	0,317
70	—0,628	0,230
75	—0,659	0,138
80	—0,674	0,043
85	—0,672	—0,054
90	—0,654	—0,149

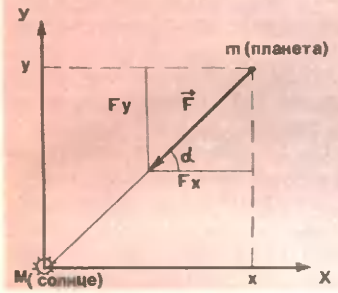
По вычисленным координатам учащиеся строят часть орбиты планеты (рис. 2). Остаток можно достроить из соображений симметрии. Затем им предлагается проверить выполнимость законов Кеплера для этой орбиты. При наличии времени интересно проследить за изменением формы орбиты в зависимости от начальной скорости планеты.

Задание 2. Вычислить абсолютные звездные величины и светимости некоторых ярких звезд, видимых в СССР, и нанести их на диаграмму Герцшпрунга—Рессела. Данные о ярких звездах взять из приложения IV учебника.

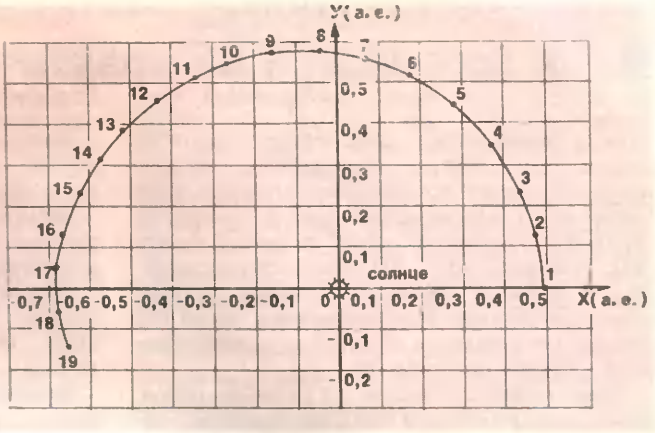
Р е ш е н и е. Формулы вычисления абсолютной величины M и светимости L по видимой звездной величине m и расстоянии до звезды в парсеках D известны учащимся:

$$M=m+5(1+\lg D),\ L=2{,}512^{5-M}.$$

Вычислив искомые значения (соответствующая программа не приводится ввиду ее



1



2

простоты) и используя данные из приложения IV учебника «Астрономия-10», учащиеся наносят яркие звезды на диаграмму Герцшпрунга—Рессела, помещенную в учебнике. Выполняя задание, они начинают лучше осознавать, как была построена эта важнейшая диаграмма, а также выявляют принадлежность ярких светил к разным последовательностям.

Задание 3. Оценить радиусы αВозничего, αБ. Пса, αБлизнецов, αМ. Пса, βБлизнецов, αВолопаса, αЛиры, αЛебедя и αЮжной Рыбы. Использовать результаты выполнения предыдущего задания и приложение IV учебника.

Размеры большинства звезд оценивают по их эффективной температуре и болометрической светимости. Последняя рассчитывается с учетом болометрических поправок.

Поскольку понятие болометрической светимости в школьном курсе астрономии не изучается, целесообразно сказать учащимся, что для определения радиуса звезды необходимо знать полную энергию, излучаемую звездой в единицу времени со всей ее поверхности. Звезды спектральных классов A, F, G, K наибольшую долю своей энергии излучают в видимой области спектра. Поэтому для приближенной оценки размеров звезд этих спектральных классов можно воспользоваться значениями мощности излучения или световой энергии, т. е. использовать значения их светимостей L. Обратившись к приложению IV учебника, учащиеся убеждаются, что высказанное допущение применимо ко всем звездам, указанным в задании. Для расчета радиуса звезды r в радиусах Солнца они выводят и используют соотношение $r = \sqrt{L}(T/T_0)^2$, где T и T_0 — температуры звезды и Солнца соответственно. Программа вычислений на ПМК не приводится ввиду ее простоты.

После расчета радиусов светил следует предложить учащимся провести устную грубую оценку объемов звезд (в объемах Солнца). Такая «прикидка» придает результатам особую наглядность. Учащиеся увидят, что среди девяти звезд, для которых они определили радиус, есть сверхгигант — αЛебедя, в сотни тысяч раз по объему превосходящий Солнце; есть гиганты и карлики. Сравнивая затем размеры звезд, их светимости и положения на диаграмме Герцшпрунга—Рессела, они убеждаются, что гиганты и сверхгиганты по светимости оказываются гигантами и сверхгигантами по размерам. Это обсуждение поможет раскрыть значение диаграммы Герцшпрунга—Рессела как диаграммы, связывающей основные физические характеристики звезд.

Результаты, полученные при выполнении заданий 2 и 3, используются при обсуждении эволюции звезд, их возраста. Просматривая свои таблицы, учащиеся смогут приводить конкретные примеры звезд, находящихся на разных стадиях эволюции.

При наличии в школе «Учебных звездных атласов» А. Д. Марленского полезно предложить учащимся сравнить рассчитанные ими светимости и радиусы звезд с данными атласа. Хорошее совпадение результатов убеждает в практической ценности полученных значений.

Использование ПМК на уроках астрономии открывает новые возможности для преодоления описательного характера преподавания предмета и увеличения доли самостоятельной работы учащихся.

Н. ГОЛАВСКАЯ, Г. САПРЫКИНА

Язык ассемблера на практических занятиях

Для организации практических занятий школьников нужно использовать любую вычислительную технику. Большую помощь в этом могут оказать шефские организации. Редкое предприятие сейчас не имеет своего ВЦ. Конечно, такие ВЦ чаще всего оснащены большими ЭВМ. Как лучше организовать обучение с использованием такой ВТ, чтобы не слишком отойти от существующей учебной программы?

Нам кажется, что в IX классе основной упор в курсе ОИВТ необходимо делать на изучение алгоритмов и алгоритмического языка. В данной работе приводится программа занятий для учащихся X класса, рассчитанная на 36 ч, основанная на использовании языка ассемблера на ЭВМ ЕС-1022. При работе с языками высокого уровня учащиеся не получают наглядного представления об архитектуре ЭВМ, хранении, преобразовании и передаче информации. Эти проблемы можно восполнить, изучая язык ассемблера. Занятия проходят 1 раз в неделю (одно занятие — это школьный урок) по следующей программе:

Занятие 1. История развития информатики и ВТ. Их роль в современном обществе. Общая структура ЭВМ.

Занятие 2. Представление информации в ЭВМ. Системы счисления: двоичная, десятичная, шестнадцатеричная.

Занятие 3. Перевод из одной системы в другую.

Занятие 4. Зачет.

Занятие 5. Представление текстовой информации в ЭВМ. Введение в основы программирования.

Занятия 6—9. Языки программирования (краткая характеристика).

Занятие 10. Экскурсия на ВЦ.

Занятия 11—12. Язык ассемблера. Средства ОС для выполнения программ на языке ассемблера.

Занятие 13. Реализация линейных алгоритмов. Составление простейших программ и управляющего пакета.

Занятия 14—15. Подготовка к работе на ЕС-1022. Этапы выполнения задания на ЕС-1022. Отладка программы. Листинг программы.

Занятия 16—18. Реализация разветвляющихся алгоритмов на языке ассемблера. Логические команды. Составление программ с использованием логических команд.

Занятия 19—22. Практические занятия на ЭВМ ЕС-1022 по отладке программ, реализующих разветвляющиеся алгоритмы.

Занятия 23—24. Организация вывода результатов счета на печатающее устройство.

Занятия 25—26. Практические занятия по отладке программ с организацией файла печати.

Занятия 27—28. Возможности адресации в языке ассемблера.

Занятия 29—30. Составление программ на языке ассемблера. Самостоятельная работа.

Занятия 31—36. Практические занятия на ВЦ.

В процессе работы на ЭВМ выяснилось, что учащиеся, несмотря на отсутствие игровых элементов в программах, работали с большим энтузиазмом, затрачивая на отладку программ внеурочное время. С большой охотой осваивается работа с магнитофоном, АЦПУ. Опыт работы по данной программе занятий показал, что за два года изучения курса ОИВТ учащиеся получают представление об использовании вычислительной техники, приобретая при этом начальные навыки работы по специальности оператора ЭВМ.

Т. ЕФИМОВА, Л. КУЛАГИНА
Ленинград

Со школьной скамьи в науку

Есть в Ленинграде школа № 239, недавно отметившая свое 250-летие и создавшая за это время немало хороших традиций. С начала 60-х гг. она была ориентирована на углубленное изучение физики и математики. Долгое время в школе были только IX и X классы, но в 1986 г. появился первый VII класс. Учениками школы могут стать

те ребята, которые за предыдущие годы проявили способности к математике, физике и другим точным наукам. В 1986 и 1987 гг. при наборе учеников в VII класс большое преимущество имели ребята, занимавшиеся в кружке Ленинградского дворца пионеров им. А. А. Жданова. Каждую весну в школе проводится открытая олимпиада по математике, которая позволяет выявить достойных претендентов на звание ученика 239-й школы.

Помимо общеобразовательных предметов в плане трудового обучения школьники изучают основы вычислительной техники, программирования и электроники. По окончании школы им присваивается квалификация лаборанта-программиста. В начале 60-х гг. в школе появилась техническая база для практического применения полученных знаний: установлена первая ЭВМ — «Урал-4», позднее — «Минск-22». С введением в школьную программу курса ОИВТ школа получила СМ-4, оснащенную 8 дисплеями, из них семь РИН-609 и один ВТА-2000.

В IX классе ребята знакомятся с основами алгоритмизации и программирования, учатся навыкам общения с машиной. Знания, полученные в IX классе, закрепляются на производственной практике в июне, проходящей в технических вузах города. С институтами заключается договор о проведении практики и утверждается ее программа, рассчитанная на 132 ч. Как правило, школьники распределяются в те институты, которые они выбрали для дальнейшего обучения. На практике ребята работают на различных кафедрах, имеющих ЭВМ, слушают лекции; по окончании практики пишут рефераты о проделанной работе.

С появлением в школе ЭВМ СМ-4 возникли проблемы. Необходимо было создать единую программу занятий по ОИВТ и трудовому обучению, так как эти две дисциплины преследуют одну и ту же цель. Дополнительно к разделам «Алгоритмизация», «Язык программирования», «Основы технологии программирования» и «Электроника» появилась потребность ввести раздел «Основы операционной системы реального времени». Однако выяснилось, что эта тема совершенно не освещена в учебной литературе; поэтому мы скомпоновали раздел курса «Основы ОС РВ» в доступном для школьников варианте, взяв за основу техническую документацию на ЭВМ и специальную литературу.

С обязательным введением производительного труда перед администрацией школы встала очень сложная задача. Как организовать его у нас для 16 классов? Необходимо было включить производительный труд в учебную сетку и связать его со специализацией школы.

Первой ступенью в плане профессионального обучения по специальности «лаборант-программист» является освоение операторских навыков на устройствах подготовки данных. Поэтому нашими шефами (Ленинградским научно-исследовательским радиотехническим институтом) в школе были установлены устройства подготовки данных на магнитных лентах ЕС-9004, на которых учащиеся IX классов набивают исходную информацию. Школьники работают по заказу горисполкома, создавая, например, банк данных для ГАИ и для учета жилой площади. За 1986/87 учебный год учащимися IX классов было освоено более 10 тыс. рублей (из них выдано 6185 рублей).

Производительный труд в X классах мы стараемся организовать в тех вузах, в которые учащиеся собираются поступать. Навыки, полученные в школе, помогают ребятам под руководством сотрудников кафедр решать достаточно серьезные научные проблемы. Так, например, ребята принимают участие в таких направлениях научно-исследовательских работ Ленинградского государственного университета, как «Разработка методов расчета движения сложных механических объектов», «Разработка методов решения задач разрушения материалов и конструкций», «Автоматизация измерений характеристик лазерного луча» и т. д. Трудовое обучение и производительный труд организованы по индивидуальным планам, с учетом склонностей и возможностей учащихся. Формой отчетности является индивидуальный план-график, который контролируют научный руководитель и учитель трудового обучения. Результаты работы оформляются в виде рефератов или отчетов. Неформальный подход к профессиональному обучению со стороны администрации школы и вузов помогает гражданскому воспитанию молодого человека, который уже в школьные годы начинает осознавать свои возможности в решении современных научно-технических задач.

Итоги деятельности школы подводятся на традиционном празднике День науки, который проводится раз в году, в феврале. В программу праздника обязательно входят встречи с ведущими учеными города, которые рассказывают школьникам о современных проблемах науки и техники. Проводится конференция с организацией секций соответственно преподаваемым в школе дисциплинам. Школьники выступают с докладами по темам, изучаемым факультативно. Финал Дня науки — концерт лауреатов фестиваля искусств и награждение авторов лучших докладов. Характерно, что на протяжении многих лет самой массовой является секция

программирования, как по количеству слушателей, так и выступающих. Лучшим докладом на одной из конференций по общему признанию стала работа В. Бреля, ученика X класса: «Экранный редактор для терминала РИН-609». Наличие такой системной программы позволило обучать школьников более доступному экранному методу редактирования информации. До этой разработки такая возможность для терминалов РИН-609 отсутствовала.

В нашей школе в рамках производительного труда и трудового обучения практикуется вовлечение старшеклассников в разработку лабораторных работ и учебно-вспомогательных программ, которые используются в школе. В 1987 г. работа «Экзаменатор», содержащая комплекс обучающих и контролирующих программ, была удостоена диплома I степени на Всесоюзной конференции школьников по основам информатики и вычислительной техники. Второй год эксплуатируется программа для обучения работе с клавиатурой, разработанная учеником А. Озолом. Разрабатывается комплекс вспомогательных программ для оперативной обработки учебно-хозяйственной информации.

Давней хорошей традицией являются школьные олимпиады по различным предметам, в том числе по программированию. В последние годы, в связи с введением в школьную программу курса основ информатики, школьная олимпиада перешла в новое качество. В Ленинграде стали проводиться городские олимпиады по программированию. Наша же школьная олимпиада, получив права районной, посылает своих победителей на городскую. Внутри школы выявляются победители в личном, а также в командном (по классам) первенстве. В 1987 г. впервые проводилась Международная олимпиада по программированию в г. Братиславе (ЧССР). Два ученика (Н. Беленький и А. Ващилло) вошли в состав команды Советского Союза.

Однако в нашей работе имеются и сложности. Работа в мультипрограммном режиме осложняет процесс обучения, так как информация, с которой идет работа, очень уязвима, и возникает необходимость наращивать средства защиты, а это, в свою очередь, ограничивает возможности учащихся и тормозит развитие индивидуальных навыков. Кроме того, эксплуатируемый в нашей школе вычислительный комплекс не укомплектован внешними устройствами, не говоря уже о магнитных носителях.

Дальнейшее совершенствование обучения мы видим в обеспечении школы техникой, более подходящей для организации учебного процесса (классом современных персональных компьютеров), а также в совершенствовании методики преподавания.

Возврат к «поштучному» производству?

Микросхемы совершили революцию в электронике. Десятки-другой операций — и на кремниевой пластине сформирована сотня схем, а в каждой из них — тысячи и тысячи транзисторов. Но в будущем нас, возможно, ожидает возврат к кропотливому изготовлению отдельных элементов, тем более кропотливому, что они будут иметь размеры порядка 10^{-9} м. Представьте себе микросхему с миллионом миллионов транзисторов!

Речь идет о нанотехнологии. В ней элементы схем формируются благодаря переносу электрическим полем отдельных атомов с нагретой иглы на подложку. Игла, конечно, особая — диаметр ее острия меньше, чем у швейной, в десятки тысяч раз. Специальные двигатели точного позиционирования способны устанавливать ее в нужное место с точностью до одного нанометра (т. е. 10^{-9} м).

Возможности, как видите, очень большие (или правильнее будет сказать — чрезвычайно миниатюрные?), но «поштучность»... Чтобы изготовить одну супермикросхему, современной большой ЭВМ придется манипулировать иглой в течение часа. Это, конечно, не промышленные масштабы. Однако нанотехнология сможет, по-видимому, «вытащить себя за волосы», подобно Мюнхгаузену. Построенные на ее основе ЭВМ будут иметь быстродействие порядка 10^{12} операций в секунду и смогут изготовлять наносхему (назовем ее так) буквально за мгновения.

Внеклассная работа

Е. КРЫЛОВА, М. ПЕТРОВ

III Московская олимпиада по информатике

В 1987/88 учебном году Московская олимпиада по информатике среди школьников проходила в третий раз, однако она значительно отличалась от двух предыдущих — и организационно, и качественно новым содержательным уровнем.

В первый год преподавания информатики в школе традиционной вузовской олимпиаде Московского инженерно-физического института (МИФИ) для школьников был придан статус общегородской. Школьный и районный туры не проводились, учителя к ее организации и проведению отношения не имели. На следующий год в результате совместных усилий МИФИ и МГИУУ олимпиада все-таки была проведена в два тура, однако решение об этом было принято поздно — зимой — и времени на качественную подготовку, в частности на создание методической комиссии и выработку концептуального подхода к составлению задач, не оставалось.

Подготовка к III Московской олимпиаде была начата заблаговременно. С 1987 г. МГИУУ проводит научно-методический семинар «Современная концепция компьютерного урока» для творчески работающих учителей; среди участников этого семинара нашлись энтузиасты, которые вместе с Е. С. Крыловой (методистом МГИУУ) и Б. Г. Киселевым (доцентом МИФИ) и составили методическую комиссию. Это учителя М. К. Петров (школа № 1121), В. А. Урнов (школа № 209), Е. М. Кузницкий (школа № 491), А. И. Ковнер (школа № 167), Д. Ю. Климов (спецшкола № 30). Методическая комиссия собиралась при подготовке задач и проведении городского и отборочного (на Всесоюзную олимпиаду) туров более десяти раз, ее членами были разработаны практически все олимпиадные задачи, кроме двух (4—IX—Р и 3—IX—Г), идеи которых предложили сотрудники лаборатории вычислительной математики МГУ.

Мы постарались отказаться от распространенного взгляда на обучение информатике в школе как на обучение программированию и, следовательно, от традиционного набора задач. В основу каждой задачи должна была быть положена та или иная идея, существенно относящаяся к информатике.

Предлагаемые ниже задачи, рассмотренные в совокупности, более или менее отражают нашу точку зрения на то, что такое задача по информатике, в частности на то, какими могут быть олимпиадные задачи по информатике.

Задачи каждого тура нумеровались в порядке возрастания их сложности, и учащиеся об этом знали. Оценивались же задачи разным количеством баллов, исходя из заранее определенного веса каждой, и эти веса школьникам не сообщались. «Стоимости» задач приводятся после условий; там же — процент учащихся, которые с ними справились.

Интересно, что в ряде случаев прогноз относительно сложности задачи не оправдался. Так, оказались труднее, чем предполагалось, задачи 1—IX—Р и 2—X—Р. Причины неудачи школьников в задаче 1—IX—Р, как ни странно, в плохом знании математики, хотя с нашей точки зрения (и мы при ней остались) в задаче идет речь о вещах элементарных.

Причины сложностей с задачей 2—X—Р — недостаточное владение техникой работы с массивами. Конечно, задачи подобно-

го типа не для массовой школы, но ведь это олимпиада! Наверное, на кружковых, факультативных занятиях надо заниматься со школьниками и выработкой технических навыков.

Значительную идеологическую трудность представили задачи на программирование работы исполнителей с данной системой команд (4—IX—Р, 3—X—Р, 3—IX—Г), особенно для школьников, воспитанных на Бейсике и Фокале, — самые лучшие из них просто написали на Бейсике (Фокале) программы, реализующие указанные системы команд, не сумев мыслить независимо, в терминах нового средства программирования. Это еще один весомый аргумент против примитивного понимания обучения информатике как обучения конкретно Бейсику (а такая точка зрения еще весьма распространена).

Особый подход был использован при составлении задач отборочного тура (на Всесоюзную олимпиаду), в котором приняли участие победители городского тура и олимпиады по программированию (она проводится в Москве наряду с олимпиадой по информатике). Нужно было отобрать всего троих из почти ста заведомо сильных ребят. Им были предложены задачи, оцениваемые не по признаку «решил — не решил», а по тому, как далеко продвинулся в решении: ведь нужен был результат не абсолютный, а относительный.

Интересно отметить, что победители олимпиады по информатике оказались отнюдь не слабее победителей олимпиады по программированию. Это является одной из причин, по которым в будущем году предполагается эти олимпиады объединить. Кроме того, успешное выступление на Всесоюзной олимпиаде команды Москвы (два первых места, одно второе и все три участника награждены специальными призами) тоже внушает нам уверенность в правильности выбранной позиции.

Школьный тур

IX класс

1—IX—Ш Две точки на плоскости, не лежащие на координатных осях, заданы своими координатами: A(X1, Y1) и B(X2, Y2). Проверить, лежат ли эти точки в одной координатной четверти. (3 очка.)

2—IX—Ш Ивана Александровича Хлестакова приглашали управлять департаментом. В первый день ему прислали 1000 курьеров, а в каждый последующий — в два раза больше, чем в предыдущий. Иван Александрович согласился тогда, когда к нему прибыло сразу не менее 30000 курьеров. На какой день это произошло? (Определить, не пользуясь умножением и делением: Иван Александрович с ними не в ладах.) (4 очка.)

3—IX—Ш Можно ли разменять 25 рублей десятью купюрами достоинством в 1, 3 или 5 рублей? (5 очков.)

X класс

1—X—Ш В компьютер князя Владимира поступили сведения о количестве змеевгорынычей, заполонивших землю Киевскую, и количестве голов у каждого из них. Илья Муромец срубает 3 головы в час. Сколько нужно таких богатырей, чтобы за сутки очистить от погани землю Киевскую? (3 очка.)

2—X—Ш Три точки на плоскости заданы своими координатами: A(X1, Y1), B(X2, Y2), C(X3, Y3). Определить, лежат ли они на одной прямой. (4 очка.)

3—X—Ш Из двух упорядоченных (в порядке возрастания) линейных таблиц размерности 1: N сформировать одну линейную таблицу, также упорядоченную по возрастанию, использовав упорядоченность исходных таблиц. (5 очков.)

Районный тур

IX класс

1—IX—Р Найти наименьшее и наибольшее значения квадратичной функции на заданном интервале. (4 очка, 40,6 %.)

2—IX—Р Малыш и Карлсон живут в прямоугольной комнате A×B. Как им посчитать, сколько понадобится квадратных ковриков со стороной C, чтобы полностью покрыть пол комнаты? (Ни Малыш, ни Карлсон не умеют ни делить, ни умножать.) Разработать соответствующий алгоритм. (6 очков, 44,6 %.)

3—IX—Р Существует ли четырехзначное натуральное число, куб суммы цифр которого равен ему самому? Разработать алгоритм построения наименьшего такого числа. (8 очков, 47,3 %.)

4—IX—Р Робот стоит перед западным концом идущего на восток клетчатого коридора неизвестной длины шириной в одну клетку. Некоторые клетки коридора закрашены. Программой для робота является последовательность занумерованных команд из следующего набора:

ШАГ НА ВОСТОК — смещение на 1 клетку;

ШАГ НА ЗАПАД — смещение на 1 клетку;

ЗАКРАСИТЬ КЛЕТКУ;

КЛЕТКА ЗАКРАШЕНА? (N) — переход

к команде с номером N при выполнении этого условия;

КЛЕТКА НЕ ЗАКРАШЕНА? (N) — переход к команде с номером N при выполнении этого условия;

КОНЕЦ КОРИДОРА? — остановка робота при выполнении этого условия.

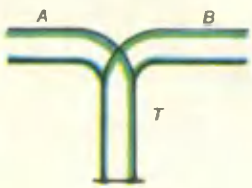
Составить программу для робота, после выполнения которой он дополнительно закрасит все те клетки внутри коридора, которые граничат с клетками, закрашенными до начала исполнения программы, и остановится у восточного конца коридора. (10 очков, 32,1 %.)

X класс

1—X—Р Произвольный выпуклый многоугольник на плоскости задан координатами своих вершин (вершины указаны по часовой стрелке). Определить количество прямых углов многоугольника. (4 очка, 56,1 %.)

2—X—Р Во время соревнования штангистов одной весовой категории в первую строку таблицы заносятся результаты, показанные в толчке, во вторую — в рывке, а в третью — собственный вес спортсменов. При равенстве результатов победитель выбирается среди штангистов с меньшим собственным весом. Напишите алгоритм, определяющий номера чемпионов по каждому упражнению и в общем зачете. (6 очков, 40,4 %.)

3—X—Р На одноколейной магистрали имеется тупик (см. рис.), попасть в который можно через две стрелки, расположенные до и после него. Длина состава — 20 вагонов, в тупике умещаются 11 вагонов, включая паровоз.



Обозначения: участок до тупика — A, после тупика — B, тупик — T; паровоз — S, вагоны — W1,...W20.

Диспетчер может использовать следующие команды:

СТРЕЛКИ В X (X — это T или AB) — переключение обеих стрелок на тупик T или на магистраль AB;

РАСЦЕПИТЬ X И Y (X, Y — это S или W_i) — расцепить, если это возможно, X и Y;

ПЕРЕМЕСТИТЬСЯ В X (X — это T, A или B) — переместить S вместе с прицепленными вагонами в X;

СЦЕПИТЬ X И Y.

Разрешается использовать процедуры (подпрограммы), составленные из указанных команд.

Напишите программу действий диспетчера, обеспечивающих перестановку двух последних вагонов поезда в голову состава. Постарайтесь использовать наименьшее число команд. (6 очков, 33,8 %.)

4—X—Р Даны две строки. Выделить максимальную общую подстроку. (10 очков, 21,3 %.)

Городской тур

IX класс

1—IX—Г Выясните, в какой координатной четверти расположен треугольник, образованный прямой, заданной уравнением У=A*X+B, и осями координат. (3 очка, 76,2 %.)

2—IX—Г Найти все трехзначные числа, равные среднему арифметическому чисел, полученных из каждого такого числа всеми перестановками (включая тождественную) его цифр. (5 очков, 59,5 %.)

3—IX—Г Из Труляляйска в Траляляйск ведет прямой туннель с двухрядным движением. Шпунтик изобрел Робота, который может двигаться на север, юг, запад, восток на ширину ряда туннеля (она равна 1 м). Траляляйск на N метров севернее и на K метров восточнее Труляляйска, причем N>K>0. Напишите программу для Робота, по которой он пройдет из центра Труляляйска в центр Траляляйска, не натолкнувшись на стены туннеля. (С давних времен центр Труляляйска и центр Траляляйска расположены на средней линии туннеля.)

При написании программы разрешается использовать команды робота (шаг на запад, шаг на восток, шаг на север, шаг на юг), все управляющие конструкции алгоритмического языка, а также арифметические операции (+, —, /, *) и команду присваивания. (6 очков, 24,6 %.)

4—IX—Г На квадратном листе клетчатой бумаги размером 30×30 в каждой клетке записано одно из чисел от 0 до 7. Если числа в двух клетках с общей стороной отличаются ровно на 1, то в обе клетки записывается большее из них. На каждом шаге производится сравнение ровно двух клеток, начинается процедура с верхней левой клетки.

а) Разработать алгоритм, который осуществляет описанную процедуру, пока это возможно.

б) Зависит ли результат от порядка перебора клеток?

в) Конечен ли этот процесс? (10 очков= =5+2+3, 23,8 %.)

X класс

1—X—Г В заданном тексте замените все последовательности повторяющихся символов на один символ и их количество, заключенное в скобки, например: АБГГГА — АБГ (3) А. (4 очка, 46,2 %.)

2—X—Г Режиссер готовится ставить новую пьесу, в которой есть 5 ролей для молодых актрис (первая роль самая престижная, а пятая — наименее привлекательная из всех). В труппе театра N молодых актрис (N>5), возможности каждой актрисы сыграть каждую роль можно оценить по пятибалльной системе. При прочих равных условиях режиссер учитывает свое отношение к актрисе, а ко всем он относится по-разному. Ту, к которой он относится лучше всех, нельзя оставить без роли, но и на первую роль назначить нельзя во избежание сплетен. По этой же причине нельзя оставить без роли ту, к которой он относится хуже всех. Режиссер решил применить для распределения ролей компьютер. Помогите ему в этом! (6 очков, 13,2 %.)

3—X—Г Произвольный выпуклый многоугольник на плоскости задан координатами своих вершин. Найти самую длинную диагональ данного многоугольника, если порядок вершин не указан, но известно, что существует только один выпуклый многоугольник с указанным набором вершин. (8 очков, 42,9 %.)

4—X—Г Пусть детектор элементарных частиц, связанный с ЭВМ, работает так, что его состояние фиксируется в ячейке памяти ЭВМ следующим образом:

исходное состояние — 0;

состояние после пролета частицы — 1. Отдельный детектор перекрывает область, которая в плане может быть представлена квадратом. При этом детектор фиксирует только попадание частицы во внутреннюю область квадрата.

Для проведения физического эксперимента детекторы сгруппировали в квадратную матрицу (наподобие шахматной доски). Перекрытая детекторами область отображается в памяти ЭВМ как двухмерная таблица, содержимое которой представляет собой цифровую фотографию исследуемой области.

Разработать программу, определяющую, содержит ли данная цифровая фотография информацию, которая может интерпретироваться как след прямолинейной траектории, начинающейся и заканчивающейся за пределами фотографии.

Допущения:

1) детектор обязательно фиксирует пролет частицы;

2) частица влетает в исследуемую область и покидает ее через центральную точку соответствующих детекторов.

(10 очков, 1 %.)

Отборочный тур

1—О Предлагается алгоритмический язык «Кобра».

В языке разрешены переменные и константы двух типов: натурального (т. е. неотрицательные целые числа) и булевского (т. е. TRUE (истина) и FALSE (ложь)), которые объявляют в начале процедуры. Пример: NAT (I, J, K) и BOOL (LOG, F) объявляют натуральные I, J, K и булевские LOG и F. Единственный выполняемый оператор — вызов процедуры по имени. Главная программа заканчивается по оператору END.

Имеется 4 стандартные процедуры.

IF (P, S1, S2) соответствует «если P, то S1, иначе S2», где P — булевская переменная, а S1, S2 — вызовы процедур.

INC(A) увеличивает натуральную переменную A на 1.

MOVE (A, B) присваивает переменной A значение переменной B, если они одного типа.

EQU (A, B, C) присваивает булевской переменной C значение TRUE, если A=B, и FALSE в противном случае.

Пользователь может вводить свои процедуры, начинающиеся с заголовка *PROC имя (список параметров)*. Процедуры допускают рекурсию.

Все переменные — локальные, все имена — произвольные последовательности латинских букв.

Пример. Процедура PLUS складывает X и Y, помещая результат в Z.

```
PROC PLUS (X, Y, Z)
NAT (X, Y, Z)
PLUS1 (X, Y, O, Z)
END
PROC PLUS1 (X, Y, T, Z)
NAT (X, Y, T, Z)
BOLL (FLAG)
EQU (X, T, FLAG)
IF (FLAG, MOVE (Z, Y), PLUS1 (X,
INC (Y), INC (T), Z))
```

Реализуйте на языке «Кобра» процедуры вычитания, умножения, деления с остатком и возведения в степень для переменных типа «натуральный». (6 очков.)

2—О На плоскости заданы N точек с

координатами (X(1), Y(1))..., (X(N), Y(N)).

а) Разработайте алгоритм соединения их непересекающейся замкнутой ломаной (если это возможно). В результате его выполнения должна быть получена таблица I[1],...I[N], указывающая порядок соединения точек.

б) Покажите, что ваш алгоритм дает правильное решение. (12 очков=7+5.)

3—О Вы организуете кооператив «Книгообмен» и имеете в своем распоряжении компьютер с достаточно большой памятью и богатым выбором языков программирования.

Создайте систему по обслуживанию клиентов, т. е. людей, которые имеют одну книгу, а хотят получить за нее какую-нибудь другую из желаемых.

На вход системы поступают данные о клиенте, его книге и его пожеланиях, после чего он становится на очередь. Клиент платит только тогда, когда найдет прямой или промежуточный обмен. Некоторые клиенты требуют немедленного обмена. Для них при невозможности обмена система предлагает информацию о всех возможных обменах его книг и книг, им заказанных.

Для создания системы вы должны выполнить работу в следующем порядке.

1. Продумайте и опишите ваш диалог с системой и все ее реакции (т. е. внешние спецификации).

2. Опишите внутреннее представление (т. е. структуру данных).

3. Разработайте алгоритмы поиска по различным запросам (например, по названию книги, ее автору, году издания, по автору книги и году ее издания и т. д.).

4. Напишите собственно программу, под управлением которой функционирует система.

(20 очков=10+5+3+2.)

Несколько примеров

По разным оценкам, в США сейчас используется от одной до трех тысяч экспертных систем (ЭС). Разрабатываемые с целью ускорения и удешевления процедуры принятия решений ЭС имеют очень широкий спектр применений. Вот только некоторые из них.

Финансовая фирма «Американ экспресс» использует ЭС для быстрого решения, предоставить ли кредит своему клиенту, если его просьба выходит за обычные рамки.

Спецслужбы США используют для борьбы с терроризмом ЭС, которая, как сообщалось, помогла предсказать одно из нападений террористов в Западной Европе.

В случае сложной неисправности автомобиля продавцы компании «Форд» скоро будут обращаться не к эксперту компании, а к ЭС.

Только ЭС ДЕФТ, «руководящая» наладкой сошедшей с конвейера продукции, дает фирме ИБМ 12 млн. долларов ежегодной экономии. Сама система стоит 100 тыс. долларов.

Фирма «Ксерокс» оснастила выпускаемые ею машины датчиками, сигналы которых передаются в ЭС РИК. Анализируя эти данные, РИК «чувствует», когда

приближается поломка, и дает дежурному ремонтнику рекомендации о мерах предотвращения аварии.

Диагностическая и лечебная ЭС «Мицин» в одном из испытаний превзошла опытных врачей, соревновавшихся с ней.

Компьютер «узнает» человека

Вряд ли кому-нибудь приходилось предъявлять удостоверение личности во время подводной прогулки с аквалангом. Но если это понадобилось бы сделать, то бумажный документ наверняка утратил бы после этого свою первоначальную свежесть.

А вот ударопрочные идентификационные карточки фирмы «Кодак» не боятся влаги, пыли, химически стойки и имеют высококачественное цветное изображение. Кроме того, на карточку нанесен код, в котором содержится информация о ее владельце. Этот код легко распознается компьютером, который может заменить и швейцара, проверяющего пропуска, и полицейского, штрафующего нарушителя, и налогового агента.

Устройство для изготовления идентификационных карточек состоит из видеокамеры, блока печати и электронного блока сопряжения с компьютером. Это последнее качество и превращает его из обычного фотоаппарата в банк данных, хотя для простого запоминания было бы достаточно видеомагнитофона.

Вся процедура изготовления карточки отнимает у клиента не более полутора минут. За это время его изображение и необходимые данные переносятся на пластиковую карточку и одновременно запоминаются в памяти компьютера. При необходимости система в любой момент готова выдать новую карточку без повторного «фотографирования», рассортировать имеющиеся в памяти данные в необходимом порядке, отредактировать их, подобрать кандидатуры по определенному признаку.

Форма и размер карточки определяются заказчиком. Если у вас нет определенных требований на этот счет, фирма предложит наиболее удобную, по ее мнению, карточку.

«Экос», «Электроника», «Бэй Фан»...

ИНФО
РЕПОРТАЖ НОМЕРА

Осенью прошлого года научно-техническое творчество молодежи в очередной раз было представлено на ВДНХ СССР. И хотя экспонаты в межотраслевом павильоне № 2 — только из Москвы и Московской области, рассказу даже о десятой их части пришлось бы посвятить номер журнала целиком. Поэтому мы выбрали пять экспонатов, во-первых, компьютерных, во-вторых, дающих представление об общем «настрое» экспозиции, о том, что могут найти

Компьютерный центр: вид «с заднего крыльца»

В компьютерном центре «Юность», работающем на выставке, все места, как обычно, заняты.

на выставке заинтересованные граждане и организации.

Среди разработок, представленных ЦООНТИ «Экос»,— персональный автоматический координатограф и имитатор стохастических изображений. Координатографом называется устройство для перемещения инструмента в нужные точки рабочего поля. На выставке он работал как графопостроитель, перемещая карандаш. Пульт управления позволяет задать электронную схему или чертеж; микропроцессорная часть обеспечивает вычерчивание заданного

на листе бумаги большого формата (до А2). Максимальная ошибка не превышает 0,05 мм. Автор этой разработки — В. В. Щелинский.

Имитатор стохастических изображений предназначен для генерирования «случайных» телекадров в реальном масштабе времени. Эта задача не так проста, как кажется. Чтобы смоделировать «белый шум» на экране телевизора, микропроцессор должен выдавать почти миллион случайных чисел в секунду! Это — быстродействие больших ЭВМ. В экспонате же, разработанном Э. П. Арзуманяном, А. И. Книжником, электроника занимает совсем немного места, однако способна моделировать кадр с 256 градациями яркости и регулируемым коэффициентом корреляции между отдельными пикселами.

Пакет программ «Автоматизированные рабочие места работников аппарата управления треста» (АРМ «Трест») рассчитан на пользователей, не имеющих опыта работы с ЭВМ, обеспечивает повышение эффективности функционирования строительных подразделений за счет оперативного учета и анализа материальных и трудовых затрат, планирования и прогнозирования строительства. В Государственном союзном сварочно-монтажном тресте внедрение первой очереди АРМ «Трест» обеспечило годовой экономический эффект 42 тыс. рублей, а всех компонентов пакета — 387 тыс. рублей. Пакет может работать на ПЭВМ «Искра-226», «Искра-1030», «Электроника-85»; изготовитель — Московский институт нефти и газа им. И. М. Губкина.

Представленный ЦНИИ «Электроника» комплекс программ для разработки принципиальных схем электронных устройств (авторы А. Ю. Кромин, И. А. Булыгин) интересен тем, что использует лишь 56К байт ОЗУ и может работать даже на ДВК-1.

На этом стенде демонстрируется комплекс программ для разработки принципиальных схем электронных устройств и имитатор стохастических изображений.

ЭВМ BF386: 32-разрядный процессор, два гибких диска по 1,2 М байт, винчестер на 60 М байт, цветной монитор 800×600 точек.

Во время счета обмен с внешней памятью отсутствует, благодаря чему достигается высокая скорость вычислений. Комплекс позволяет разрабатывать устройства с общим числом выводов у комплектующих изделий на одном субблоке до 2000 и устройства с таким же числом контактов на разъемах входящих в него субблоков, выдает всю необходимую конструкторскую документацию вплоть до файлов для автоматической сверловки и трассировки печатных плат.

И наконец, «гвоздь» компьютерной части выставки — стенд кооператива «БФ компьютер — Москва». Это монопольный представитель в СССР китайской фирмы «Бэй Фан компьютер корпорейшн», самой известной моделью которой является 32-разрядная ПЭВМ BF386, выпущенная всего через две недели после появления на рынке аналогичной модели серии PS/2 фирмы ИБМ. Производит «Бэй Фан» и модели, разъемно совместимые с IBM PC. Кооператив же создан для оказания посреднических услуг и организации технического обслуживания оборудования фирмы, проданного в СССР. Кроме того, на выставке демонстрируются и проверяются на совместимость с IBM PC программы, разработанные в СССР, с целью обеспечения возможности их экспорта. Еще одна цель кооператива — обеспечение молодежных научно-технических коллективов современной вычислительной техникой в рамках «Программы мобилизации интеллектуального потенциала страны», разработанной ЦК ВЛКСМ.

ВНИМАНИЮ ЧИТАТЕЛЕЙ

Киевский филиал Института повышения квалификации руководящих работников и специалистов Министерства промышленности средств связи СССР проводит переподготовку руководящих работников и специалистов (в том числе преподавателей учебных заведений) по следующим специализациям.

1. На базе ВНИИВЦ «ТОН» (Всесоюзный научно-исследовательский внедренческий центр по организации труда и управлению производством «ТОН»).

1.1. Управление трудоемкостью изделий на предприятии; организация учета и планирование снижения трудоемкости. Расчет трудоемкости с применением ЭВМ.

1.2. Нормирование труда рабочих, занятых выполнением механозаготовительных работ.

1.3. Нормирование труда рабочих, занятых выполнением сборочно-монтажных работ.

1.4. Организация труда специалистов НИИ, КБ.

1.5. Коллективные формы организации труда на предприятии.

1.6. Материальное стимулирование труда работников предприятия.

1.7. Организационное управление предприятием, объединением.

1.8. Совершенствование организации труда на рабочих местах работников предприятий на основе аттестации.

1.9. Организация социалистического соревнования на предприятии.

2. На базе ПО им. С. П. Королева.

2.1. АРМ пользователя на базе персональных ЭВМ Нейрон И9.66 (эксплуатация и ремонт).

2.2. Использование ПЭВМ пользователями-непрофессионалами.

2.3. АРМ разработчика СВТ на базе систем отладки типа СО-05 (Нейрон И3.68).

2.4. Базовое программное обеспечение ПЭВМ Нейрон И9.66.

2.5. Операционная система ДОС1 (MS-DOS).

2.6. Операционная система ИНМОС (INMOS).

2.7. Микропроцессорные средства вычислительной техники (МПК К-1810).

2.8. Локальная сеть ПЭВМ Нейрон ЛВС (Нейрон И9.66 — СМ 1420).

3. На базе Межотраслевого научного центра технологии программирования и Института кибернетики им. В. М. Глушкова АН УССР.

3.1. Технология программирования на ЭВМ типа ДВК-2, 3.

3.2. Технология программирования на ЭВМ типа IBM PC (Нейрон, ЕС-1840).

3.3. Технология программирования на ЭВМ типа «Электроника-85».

3.4. Технология программирования для мини-ЭВМ типа СМ-4.

3.5. Технология программирования на ЭВМ типа «VAX» (Изот 1055 С).

3.6. Технология разработки программных средств САПР.

4. На базе НПО «Маяк».

4.1. Бытовая аппаратура магнитной звукозаписи (электрические каналы и компоненты).

4.2. Бытовая аппаратура магнитной звукозаписи (лентопротяжные механизмы и системы электропривода).

4.3. Цифровая магнитная запись.

4.4. Бытовая видеотехника.

4.5. Проектирование аппаратуры точной магнитной записи и видеозаписи.

4.6. Обеспечение надежности БРЭА.

4.7. Дизайн и эргономика бытовой и специальной АМЗ.

4.8. Технология производства деталей и узлов АМЗ.

4.9. Техника и технология оптической (лазерной) записи информации.

4.10. Непрерывная подготовка по САПР АМЗ:
БАМЗЗ (электрические каналы);
БАМЗЗ (ЛПМ и САР);
ЛЦМЗ;
АТМЗ.

Обучение проводится в течение одного месяца в объеме 144 часов с отрывом от основной работы. Слушателям читаются лекции по основным вопросам каждой специализации, проводятся практические занятия, выполняется курсовая работа по теме, утвержденной руководством командирующей организации.

Лекционные занятия проводят ведущие ученые и разработчики отрасли. Аннотации программ специализаций высылаются по заявкам.

Заявки принимаются по адресу: 252065, г. Киев-65, ул. Метростроевская, 5 А, директору КФИПК В. П. Анненкову.

Справки по тел. 488-38-66 (междугородный код Киева — 044).

Журнал в журнале
для школьников, студентов,
учащихся СПТУ и техникумов

Издается при участии ЦК ВЛКСМ

МОЛОДЕЖНАЯ ИНИЦИАТИВА

1

В НОМЕРЕ:

Поздравьте своих учителей

Два самых красивых праздника в школе: День учителя и 8 Марта. Всегда хочется поздравить учителей, большинство из которых — женщины, так, чтобы это поздравление сохранилось у них в памяти до следующего праздника. Перед праздником 8 Марта члены нашего кружка вычислительной техники во главе с учащимся Х А Василием Милюшниковым предложили использовать для этой цели компьютерную технику. Идея понравилась всем. В актовом зале, где должен был проходить торжественный педсовет, установили телевизор и подключили к нему БК-0010. В течение всего ритуала поздравления учителей, вручения грамот, чествования юбиляров работал установленный нами комплект. На экране телевизора сменялись имена и отчества учителей в поздравительном тексте. А все поздравление в целом заканчивалось изображением красивого цветка.

Эффект был потрясающий. Многие учителя знали о компьютерной технике только понаслышке. И теперь они неотрывно следили за изображением на экране и очень живо и эмоционально воспринимали свое имя, отчество, появляющиеся в поздравительном тексте.

А наши кружковцы, особенно Вася, который составил программу, услышали много теплых слов благодарности за приятный сюрприз.

Текст программы, который приводится ниже, при небольшой корректировке может использоваться как поздравительный для любого праздника.

```
10.01 S L=6.3
10.03 S A=10; S B=4
10.05 S C=8; S D=3
10.10 X FCHR [12,154]
10.20 DO 90
10.30 T !! '        УВАЖАЕМАЯ'
10.40 T !!!!
10.50 T ' ПОЗДРАВЛЯЕМ ВАС С ПРАЗДНИКОМ'!
10.60 T ' ВЕСНЫ И МЕЖДУНАРОДНЫМ ДНЕМ'!
10.70 T '        8 МАРТА! ' !!
10.80 T 'ЖЕЛАЕМ КРЕПКОГО ЗДОРОВЬЯ',!!
10.90 T 'СЧАСТЬЯ, ТВОРЧЕСКИХ УСПЕХОВ'!!
20.10 T 'В ТРУДЕ, ПРЕКРАСНОГО НАСТРОЕНИЯ'!!, !!!  !!»
20.20 X FK [3,5] ; T 'ГАЛИНА АЛЕКСЕЕВНА'; DO 60
20.30 X FK [3,5] ; T 'ЕВГЕНИЯ КУЗЬМИНИЧНА' ; DO 60
20.40 X FK [3,5] ; T 'ЕЛЕНА МИХАЙЛОВНА'; DO 60
20.50 X FK [3,5] ; T 'ЛЮДМИЛА ЗАХАРОВНА'; DO 60
50.10 X FCHP[12]; DO 90
50.20 DO 70
50.30 DO 60
50.40 G
60.10 F I=1,0.  3,50;S T=FSIN[FSQT(1)]
70.10 X FT [1,128,100]
70.20 F I=0,01,L;SR=A*FSIN[R*I]+C*FCOS[D*I]; DO 80
80.10 S X=R*FSIN[I]*5
80.20 S Y=R*FCOS[I]*5
80.30 X FV [1,X+128,Y+100]
90.10 X FK [1,1]
90.20 T 'ПОЗДРАВЛЯЕТ 10А'
```

Комментарии к программе

Блоки 10—40 изображают на экране текст поздравительной открытки. Строка 10.40 оставляет на экране место для имени, отчества учителя, которые формируются блоками 20—40. После каждой строки блоков 20—40 идет обращение к строке 60.10, в которой вписан цикл задержки времени. Блоки 70—80 рисуют цветок. Программа организована таким образом, что после запуска она будет повторять текст поздравительного послания много раз (строка 50.40), до тех пор пока не отключат установку или не остановят программу клавишей «стоп».

В тексте программы для примера приведены имена и отчества только четырех учителей.

А. КАПЛУН,
СШ № 20, Донецк

Копировщик дисков

В программном обеспечении ПЭВМ «Агат» имеется ряд программ, которые предназначены для копирования дисков (целиком или по файлам — COPY, FID, отдельными треками — COPY, КОПИТРЕК). Однако они не позволяют восстанавливать программы с испорченных дисков, так как при считывании информации с диска в случае неполноценности хотя бы одного сектора вся уже считанная информация в ОЗУ уничтожается. Таким образом, минимальные потери информации исчисляются тысячами байт (минимум теряется 1 трек, содержащий 4К байт).

В созданной нами программе «Копировщик дисков» предусмотрены широ-

кие возможности. Она реализует обмен с диском на трех уровнях: поблоковом, потрековом и пофайловом, что намного облегчает труд программиста, работающего с дисками, и позволяет «выбрасывать» из трека только ошибочные сектора, уменьшая потери до сотен байт. Есть и возможность многократных попыток считывания сбойного сектора, что иногда позволяет сохранить его.

«Копировщик» позволяет также просматривать содержимое ОЗУ в текстовом виде, переводить числа из десятичной системы счисления в шестнадцатеричную и обратно, изменять информацию в ОЗУ после выхода в диалог программы «Системный монитор».

Диалог «Копировщика» с пользователем реализован по принципу меню. В любой момент пользователь может просмотреть инструкцию и определить, какой из режимов работы выгоднее выбрать в данном конкретном случае.

Данная программа отлажена и успешно используется в УПК г. Златоуста. Адрес для запросов: 456209, Златоуст, ул. Островского, 6. Телефоны для справок: 3-43-62, 3-42-27.

Э. КАЛИНИН,
А. РУБЦОВ
Златоуст, СШ № 15, X класс

Разработка студента

Сейчас много говорится и пишется о программных педагогических средствах для школ и вузов. По-моему, пока их надо создавать с тех позиций и традиций, к которым мы привыкли. Каждое занятие начинается в школе с классного журнала. Вот я и заказал такой электронный журнал студенту II курса МФТИ В. Быкову. Он создал журнал на языке Паскаль. Работа удалась.

Созданный «Электронный классный журнал» сохраняет утвердившиеся традиции ведения обычного классного журнала, а поэтому пользоваться им на ПЭВМ не составляет никакого труда. Запрос пароля при входе в программу и шифровка данных в файлах защищают информацию от несанкционированных изменений.

Такой журнал (на языке Паскаль или транслированный) можно приобрести в Мозырском городском межшкольном учебно-производственном комбинате трудового обучения и профессиональной ориентации учащихся с согласия автора.

Данный журнал создавался для ДВК-2, ДВК-3 и ДВК-4.

Наш адрес: 247760, Гомельская обл., г. Мозырь, ул. Пролетарская, 84, городской межшкольный учебно-производственный комбинат.

Г. СУВОРОВ,
мастер производственного обучения операторов ЭВМ

ТВОРЧЕСТВО НАШИХ ЧИТАТЕЛЕЙ

Одна голова — хорошо, а две-три! **Фото В. Булдакова**

Нуждается в доработке

Первые владельцы «Электроники БК-0010», разговаривающей на диковинном Фокале, недоумевали: кому понадобилось снабжать первый наш бытовой компьютер столь экзотическим языком программирования вместо Бейсика, получившего распространение на большинстве зарубежных компьютеров, или Паскаля, который в последнее время пытается отобрать у Бейсика пальму первенства! Общее удивление не осталось без внимания, и вскоре заговорили о том, что вот-вот появится Бейсик с широкими возможностями, разработанный специально для БК. Прошло не так уж много времени, и разрекламированный труд системных программистов, разместившись в трех микросхемах, занял законное место в корпусе БК—0010.01. Это была так называемая июньская 1986 г. версия Бейсика MSX, занимающая 24 К, т. е. львиную долю памяти компьютера. Казалось бы, остается только радоваться, но дотошные пользователи вскоре установили, что это всего лишь усеченный вариант февральской версии того же года, которая, кстати, занимала в памяти те же 24 К. Кто и зачем произвел хирургическую операцию — непонятно, но в результате в языке исчезли некоторые важные команды и функции. Нет, например, в новой версии команды MERGE, необходимой для объединения двух Бейсик-программ, отсутствует команда KEYLIST — просмотр текста ключей и SPC — печать пробелов. Пропала функция BF для закрашивания прямоугольника. Но, что самое удивительное, неэффективно составлен код компилятора, а значит, тот же Бейсик мог бы занимать в памяти меньше места и работать быстрее.

Об этих и других недостатках в редакцию написали москвичи Д. Баронов и И. Николаев, письма которых мы публикуем.

Можно спорить о достоинствах и недостатках языка, который хранит при исполнении исходный текст программы и представление ее в машинных кодах. Существенно то, что максимальный объем программы, которую можно запустить,— всего 7 К, причем без учета данных. Может быть, для БК-0011 это и несущественно, но для БК-0010 такого остатка памяти явно недостаточно. По-моему, стоило бы присмотреться к версии Бейсик-87, которая загружается с магнитной ленты, занимает меньше памяти и почти не уступает «штатному» Бейсику в быстродействии. Дополнительные трудности с памятью сегодня связаны еще и с тем, что в одной строке программы записывается лишь по одному оператору, а номер каждой строки также занимает память.

Как ни странно, чрезвычайно медленно работают символьные операторы и функции. Как вы думаете, сколько секунд может выполняться программа (рис. 1)? Секунду? Две?

```
 5 CLEAR 5200
10 DIM T$(255)
20 FOR I=0 TO 255
30 T$(I)=STRING$(20, 32)
40 NEXT I
50 END
```

Почти пятьсот секунд!

Удивительным образом работает функция STR ¤ Вместо того чтобы превращать числа в строки символов, она к тому же добавляет слева и справа пробелы. Проверьте, чему будет равно LEN (x ¤) в программе (рис. 2).

```
10 X%=5
20 X$=STR$(X%)
30 PRINT LEN(X$)
40 END
```

Единице? Нет, трем — вычисляет БК.

Что-то произошло с многомерными множествами. Если указывать переменную с тремя или более индексами, БК сочтет их одним символом, эквивалентным их сумме. Не верите? Проверьте! (Рис. 3.)

```
  DIM P%(2,2,2)
OK
  LET P%(1,1,1)+5
OK
  PRINT P%(0,2,1)
5
OK
  PRINT P%(2,0,1)
5
OK
```

Если вы даже разберетесь с тем, как определить подпрограмму или функцию в кодах с помощью команды DEF USR, вы все равно не сумеете записать эту подпрограмму на магнитофон или считать ее с ленты.

Есть, наверное, и другие

ошибки, до которых мне пока что не удалось докопаться. Впрочем, вывод можно сделать уже сейчас — версия Бейсика для БК нуждается в серьезной доработке.

Д. БАРОНОВ, Москва

Таблица 1

```
10700 10037 2030 162737    10   2030 62700  26
10037    34  137 134612 21627 136116  1402 137
146404    2
```

Хотя Бейсик вместе со своими недостатками, как говорится, зашит в ПЗУ, его можно усовершенствовать, точнее, приблизить к стандартной версии, исправив некоторые ошибки. Исчезнувшую было команду MERGE можно реализовать на основе «штатной» команды LOAD. Если выполнить эту команду тогда, когда в памяти БК уже имеется одна программа на Бейсике, машина тотчас же выдаст сообщение об ошибке 5. Это сработает команда TRAP 5, расположенная в ячейке 136114. Достаточно заблокировать эту ячейку, и LOAD превратится в MERGE, тогда новую программу можно будет объединять со старой. При этом важно запомнить, что из двух бейсичных строчек с одинаковыми номерами остается та, что принадлежит вновь считываемой программе. Текст программы, которая превращает LOAD в MERGE,— в табл. 1. Принцип ее действия весьма прост. Каждое прерывание по команде TRAP передается на подпрограмму. Если адрес команды равен 136114, прерывание будет заблокировано, а управление вернется к Бейсику.

Программу набирают в режиме отладки, но, если блока МСТД под рукой нет, ее можно загрузить и из Бейсика (рис. 4). Чтобы не набирать ее каждый раз с клавиатуры, запишите программу на магнитофон командой BSAVE «M», чтобы запускать ее можно было командой BLOAD «M», R. После того как программа загружена и запущена, разрешается выполнять любые команды Бейсика, даже NEW и CLEAR арг 1, арг 2. Единственное условие — значение второго аргумента в CLEAR должно быть меньше адреса загрузки 37720.

Программа теряет свои функции после нажатия клавиши СТОП или при выполнении команды MONIT. Чтобы вновь ее запустить, достаточно одной команды Бейсика POKE 28, арг, где арг — число, равное сумме числа 24 и адреса загрузки программы.

```
10 DATA 4544,4127,1048,—6689,8,1048,26048,22
20 DATA 4127,28,95,—18038,9111,—17330,770,95
30 DATA —13052,2
40 FOR AD=16336 TO 16370 STEP 2
50 READ X
60 POKE AD,X
70 NEXT
BSAVE «M», 16336, 16372
```

И. НИКОЛАЕВ, Москва

Программная орбита

24. Рига-микро. Система. (НИИ ФТТ ЛГУ им. П. Стучки). Т-язык.

24.1. Системная дискета — базовая операционная система для ДВК-2М РАФОС и программа обслуживания сети «Рига».

24.2. Дискета обучения работе в ОС РАФОС — набор обучающих программ для приобретения учителем навыков работы с ДВК-2М.

24.3. Дискета первого контакта — набор программ, формирующих начальные навыки работы с БК-0010.

24.4. Дискета обучения языку Бейсик — для полной или частичной автоматизации процесса обучения языку Бейсик.

24.5. Дискета изучения применения ЭВМ — для проведения занятий по использованию ЭВМ для подготовки текстов, табличных расчетов, ввода и поиска информации.

24.6. Дискета демонстрации возможностей ЭВМ — набор программ, демонстрирующих возможности ЭВМ при обучении различным общеобразовательным дисциплинам (физике, биологии, химии и т. д.).

Более подробное описание системы см.: Информатика и образование, 1988, № 2.

25. КОФОК. Кодово-фокальная система (г. Москва). Система предназначена для создания и поддержки кодово-фокальных программ, ее применение позволяет использовать практически все программное обеспечение, заложенное в ПЗУ компьютера.

Осваиваем Бейсик

Эффективность программ на Бейсике может быть значительно большей, если использовать подпрограммы в машинных кодах. Освоить работу с ними не так уж трудно, два примера таких подпрограмм предлагают наши авторы из Ленинграда. Ну а тем, кто заинтересуется тонкостями, советуем прочитать статью москвича Д. Баронова «Включение подпрограмм в машинных кодах в программы на Бейсике для БК-0010-01» (Логическое программирование. М.: Знание, 1988. Новое в жизни, науке, технике. Сер. «Вычислительная техника и ее применение»; № 9).

Первая подпрограмма, назовем ее «перевертыш», заставляет переворачиваться символы на экране БК, стоит обратиться к ней из Бейсика командой A-USR(A) Рекомендуем ее тем, кто составляет на Бейсике игровые программы. Вторая подпрограмма позволяет озвучить Бейсик. С ее помощью в программы на Бейсике можно включать фрагменты мелодий и различные звуковые эффекты. Обращение к такой подпрограмме чуть посложнее. Прежде необходимо определить какую-нибудь символьную переменную, состоящую из заглавных букв латинского алфавита и знаков препинания. Символы, стоящие на нечетных позициях, будут определять длительность звукового сигнала, а те, что стоят на четных позициях,— высоту звука. Звучат клавиши ФЫВАПРОЛДЖЭ. Определить символьную переменную можно по-разному, например оператором присваивания или оператором INPUT. Чтобы БК-0010.01 зазвучала, к подпрограмме обращаются командой A$=USR1(A$). Как объединить подпрограмму в машинных кодах с программой на Бейсике? Прежде всего ее нужно объявить оператором DEF USR0=&01000. Текст программы в кодах записывается в оператор DATA. Так как данными на этот раз являются восьмеричные числа, то перед каждым кодом ставьте &0. Перед запуском основной программы организуйте небольшой цикл FOR, который будет читать очередной код из оператора DATA и записывать его в память оператором POKE, начиная с 1000-го адреса. Для «перевертыша» все подготовительные действия уже закончены, для «музыки» необходимо определить символьную строку так, как это было рассказано раньше.

```
012701 042000 012702 100000 012703 000100 160302 010200
006203 011104 011021 010420 077304 020102 001365 000207

012503 006203 001417 011500 112002 162702 000060 010701
062701 000060 122110 001407 105721 001374 077210 122043
005703 001362 000207 005046 111116 012601 010104 012737
000100 177716 077401 010104 012737 000000 177716 077401
077213 000755 177506 170103 161531 153125 145127 137501
132105 125120 120116 113522 107507 103517 077514 074135
070504 065532 062526 057534 055072 052456 050010 045437
043422 041422 037436 000000
```

А. ПИЛИН, А. КАЗАН,
Ленинград

И снова о ключах

После публикации заметки В. Шеломанова «Ключ к Фокалу» в прошлом номере журнала редакция получила немало писем, авторы которых считают, что описание было слишком сжатым и к вопросу программирования ключей стоит вернуться. Как известно, ключ клавиатуры (или программируемые функциональные клавиши)— средство облегчить ввод информации в машину. Стоит нажать одну из цифровых клавиш и клавишу НР или АР2, и на экране появится строка заранее запрограммированных символов произвольной длины (но не более 255 символов). Это может быть командное слово, имя файла, одна или несколько строк программы и так далее. В Бейсике БК-0010.01 предусмотрена возможность запрограммировать 10 ключей, хотя максимальная длина каждого ключевого слова должна быть не более 16 символов. Однако ключи можно программировать и на БК-0010, причем непосредственно в машинных кодах.

Выполняя пожелания читателей, передаем слово Ю. Дворецкому из г. Северодвинска.

Прежде всего нужно выйти из Фокала в режим тестов,

в ответ на + нажать клавиши РУС Т С. «Солнышко» на экране означает, что мы находимся в режиме отладки. Программу для ключей будем размещать с адреса 1000. Адрес выбран не случайно — дело в том, что первую тысячу ячеек занимает так называемый системный стек — область оперативной памяти, где размещаются необходимые для работы компьютера параметры и системные переменные. Если разместить программу в какой-либо другой области ОЗУ (за небольшим исключением), то при следующем переходе в Фокал память будет автоматически очищена, а программа утеряна. Неизменным остается содержимое только тех ячеек, что расположены по адресам от 1000 до 2000. Впрочем, при возвращении в Фокал из режима отладки половина области (с 1400 по 1776) используется пусковой программой, так что для ключей остается не так уж много места — ячейки с 1000 по 1376, т. е. всего 256 байт.

Поставщиками информации для арифметико-логического устройства микропроцессора служат его регистры — своеобразные почтовые ящики, откуда поступают команды, операнды и адреса. Для программирования ключей в нулевой регистр микропроцессора нужно поместить номер ключа (от 1 до 10), а в первый регистр— адрес ячейки, в которой находится первый байт текста ключа. Здесь не обойтись без команды «пересылка». Как ею пользоваться? Код 12700 означает, что мы хотим переслать в регистр 0 содержимое ячейки памяти, которая расположена следом за той, где записан код. Вот в этой ячейке и записывают номер ключа. Аналогично код 12701 перешлет содержимое следующей ячейки (адрес ячейки, где начинается текст ключа) в регистр Р1 микропроцессора. Замыкает цепочку команда 104012 — сигнал микропроцессору на программирование ключа. Эта последовательность команд повторяется столько раз, сколько ключей мы собираемся запрограммировать, но не больше 10. Затем по указанным в тексте объявлений ключа адресам размещают основные тексты ключей. В каждой ячейке, кроме первой, записывают по два символа из текста ключа: первый символ — в первом (младшем) байте, второй — во втором (старшем) байте. Первая ячейка содержит только один символ — первый символ текста ключа в старшем байте, а в младшем необходимо указать число символов текста, т. е. длину ключа.

Хотя никаких сложностей здесь нет, поясним сказанное примером: текст ключа — слова «процессор». В нем 9 букв, т. е. 11 символов в восьмеричной системе. Буква П — первый символ, его код 320, а значит, содержимое первой ячейки памяти 011320. Далее следуют буквы Р (код 322) и О (код 317), записанные как 322317, и т. д.

Для тех, кто еще не освоил режим отладки, поясню, как вводить готовую программу. Наберите 1000 А — так вы установите текущий адрес 1000. Набирайте первую команду нашей программы — 12701 и нажимайте клавишу «,» — команда запишется по адресу 1000, текущий адрес станет равен 1002. (Кстати, такой способ записи принято называть автоинкрементным.) Вводим 1 — номер ключа и снова нажимаем клавишу «,». Затем 12701 и дважды нажимаем «,». Первое нажатие отправляет в память команду 12701, второе — дает сигнал отладчику пропустить одну ячейку (мы еще не знаем, где будут располагаться тексты ключей). Набираем команду 104012, затем «,» и так повторяем столько раз, сколько нужно ключей. Разумеется, каждый раз номер ключа должен быть другим. После ввода последней команды последнего ключа вводится еще одна команда — 0 — команда на останов. Следующее действие — ввод текстов ключей. Прежде всего нажмите клавишу А — БК сообщит вам значение текущего адреса. Выпишите его на листок бумаги. Наберите число символов первого ключа и введите его в память командой « . » . Точка вместо запятой увеличивает адрес только на единицу, так мы сможем записывать текст ключа символ за символом, байт за байтом. Вводим код первого символа, «.», код второго и «.» и т. д. После того как в памяти оказался последний код символа первого ключа, запросите текущий адрес командой А. Запишите его на ту же бумажку и вводите текст второго ключа — последовательность действий остается прежней. Когда будут введены тексты всех ключей, останется лишь проставить записанные на бумажке адреса в ячейках после команд 12701. Программа готова. Установите

12700	1	12701	1030
104012	12700	2	12701
1040	104012	0	
5373	341373	353351	
10342	351357	355341	363363
341000			

адрес 1000 (команда 1000 А) и наберите 400Л. На экране распечатается ваша программа. Проверьте, все ли правильно, и запишите ее на магнитофон — командой М3 (Адрес=1000, Длина=разность между 1000 и адресом последней ячейки программы, Имя — любое). Вот теперь программу можно запустить — 1000 G. Если в

ответ снова появилось «солнышко» — всё в порядке. В Фокал нужно выйти не по команде К, а с помощью команды 120000 G (в монитор выходить нельзя).

Добавлю, что кроме букв и цифр текст ключа может содержать коды спецсимволов, знаков препинания, математических знаков, а также коды различных команд — очистки экрана, переключения цвета и других. Вот одна замеченная особенность — для исполнения команды «ввод» в тексте ключа используют код 15 вместо указанного в «Руководстве» кода 12. В заключение — пример программирования двух ключей. Попробуйте разобраться в нем без помощи БК.

Маленькие хитрости

Один из простейших способов защитить программу от нежелательного доступа — дать ей имя несколько необычным способом, например используя различные регистры. Если нажимать клавиши в такой последовательности «ЛАТ AL РУС О Х 5 стрелка влево стрелка влево ЛАТ NI ВВОД», то на экране имя будет печататься как ALONI, однако, чтобы считать программу с ленты, к ней нужно обратиться именно так, как мы это делали при записи.

М. НУРМАГОМЕДОВ,
г. Махачкала

Неплохую заставку к своим программам на Фокале можно делать на основе фрагмента всего из четырех коротких строк. БК, действуя по такой программе, будет периодически инвертировать картинку на экране, а также выдавать звуковой сигнал до тех пор, пока кто-нибудь не нажмет любую клавишу. В этом случае управление будет передано на строку 1.30, где должна начинаться основная программа.

```
1.10 X FCHR(12,154)
1.15 S T=FX(1,177716)
1.20 I (T+32608) 1.30, 1.30, 1.25
1.25 X FCHR(7,157,7); G 1.15
1.30 Q
```

М. БУРЛУЦКИЙ,
г. Междуреченск

Гармония чисел

Как-то вечером я развлекался вычислением коэффициентов ряда Фурье. Программа была почти готова, когда у меня появилась идея наглядно представить вычисленные компоненты ряда. Так возник первый цветок Фурье. Картинка мне понравилась, и я тут же переписал программу на Фокале специально для рисования цветка. Она получилась совсем простой, поэтому я предлагаю ее без пояснений.

```
1.10 X FCHR (12)
1.20 C ПРИМЕР: N=5, A1=1, A3=0.25, A5=0.1,
     X0=110, Y0=110, R0=100.
2.10 T 'КООРДИНАТЫ ЦЕНТРА ЦВЕТКА',!
2.20 A 'X0', X0, 'Y0', Y0,!
2.30 A 'СРЕДНИЙ РАДИУС ЦВЕТКА R0', R0,!
2.40 A 'КОЛИЧЕСТВО ЛЕПЕСТКОВ N', N
2.50 T ! , 'ОТНОСИТЕЛЬНЫЕ АМПЛИТУДЫ
     1, 3 и 5 ГАРМОНИК (0...1)',!
2.60 A 'A1', A1, 'A3', A3, 'A5', A5
2.70 X FCHR (12)
2.80 S P=3.14159/180
3.10 S B1=R0*A1/(A1+A3+A5)
3.20 S B3=R0*A3/(A1+A3+A5)
3.30 S B5=R0*A5/(A1+A3+A5)
3.80 F K=0, 0.5, 359.5; D4
3.90 G 5.1
4.10 S R1=B1*FSIN(N*P*K/2)
4.20 S R3=B3*FSIN(3*N*P*K/2)
4.30 S R5=B5*FSIN(5*N*P*K/2)
4.40 S R=FABS(R1+R3+R5)
4.50 S X=X0+R*FCOS(P*K)
4.60 S Y=Y0+R*FSIN(P*K)
4.70 X FT(0, X0, Y0); X FV(1,X,Y)
4.80 S R=R+4; D4.5; D4.6
4.90 X FT(1,X,Y)
5.10 F K1=0, R0/50, R0/10; D5.3
5.20 Q
5.30 F K2=0, 15, 345; D6
6.1  X FT[0,X0+K1*FCOS(P*K2),
     Y0+K1*FSIN(P*K2)]
```

В. ВИНОГРАДОВ,
г. Клин

Точка зрения

К. ШОЛОМИЙ

О дефиците программных средств для компьютерного обучения школьным предметам

Компьютеризация школы идет, как известно, в двух направлениях: обучение компьютерной грамотности и применение компьютера для обучения школьным предметам (компьютерное обучение — КО). В развитии первого направления достигнуты известные успехи, и его перспективы выглядят достаточно оптимистично. Иначе обстоит дело в отношении КО. Здесь ситуация катастрофическая: практически нет обучающих программ, и ожидать, что их дефицит будет преодолен ко времени массового оснащения школы компьютерами, нет оснований. Говоря о дефиците обучающих программ, мы имеем в виду не любые программы, а лишь совершенные в психолого-педагогическом отношении. Программы, не удовлетворяющие этому требованию (а их подготовлено уже немало), непригодны к использованию в учебном процессе. Компьютер — чрезвычайно сильный инструмент, и при хороших программах может дать значительный педагогический результат (по американским данным, применение компьютера позволяет сократить срок обучения на одну треть [2]). Но значительным может быть и вред при плохих программах.

Утверждение о дефиците программ может показаться странным. Подготовкой программных средств для КО занимаются только по линии АПН СССР, АН СССР и бывшего Минпроса СССР не менее шести коллективов: в системе АПН это НИИ информатики и вычислительной техники[1]; в системе АН — подразделения Института проблем информатики, Института программных систем, Научно-исследовательский вычислительный центр, Временный коллектив «Школа-1»; в системе бывшего Минпроса СССР — Главный информационно-вычислительный центр. Работают по этой проблематике также ряд более мелких коллективов. Таким образом, на подготовку программных средств для КО ежегодно расходуются крупные средства. Тем не менее обучающих компьютерных программ (повторяю: пригодных для использования в учебном процессе школы) практически нет. В чем же дело, почему столь значительные затраты сил и средств не обеспечивают решение проблемы?

Причин здесь много, и главная из них состоит в том, что к ее решению привлекаются преимущественно те, кто решить эту проблему не в состоянии в силу отсутствия соответствующей компетенции, а те, кто такой компетенцией располагает, от решения отстранены. Охарактеризуем ситуацию более детально.

КО — принципиально новый вид обучения, еще не исследованный в достаточной степени наукой и не освоенный практикой. Поэтому готовых, так сказать, стопроцентных специалистов, способных на высоком психолого-педагогическом уровне и при полном использовании возможностей компьютера создавать программы для КО, у нас нет. (Нет, кстати говоря, их и за рубежом, и жалобы на недостатки существующих программных средств для КО красной нитью проходят через англо-американскую литературу, например [8], [9]). Где же выход, откуда взять требуемых специалистов?

Хорошим примером, демонстрирующим как надо действовать в подобной ситуации,

[1] Психолого-педагогический потенциал АПН СССР на официальном уровне от решения задачи создания программных средств для КО в основном отключен. Например, в планах НИИ СиМО, НИИ ОПП и НИИ ШОТСО эта проблематика отсутствует.

является подход, обеспечивавший у нас подготовку космонавтов, т. е. ранее отсутствовавших специалистов. Известно, что для этого в начальный период освоения космоса были использованы летчики, поскольку они до специального обучения в наибольшей степени обладали нужными для космонавтов знаниями, и их подготовка могла быть осуществлена за минимальное время.

Можно сказать, что и подготовка специалистов, способных создавать совершенные программы, окажется менее трудоемкой, если их будут готовить из тех, кто уже обладает в максимальной степени требуемым опытом. Именно такой опыт накоплен бывшей лабораторией программированного обучения Института психологии АПН СССР, которая до 1977 г. в течение почти пятнадцати лет была головным центром исследований в области программированного обучения. Учитывая, что КО — разновидность программированного обучения [5], можно сказать, что работа лаборатории фактически была направлена на создание теоретических основ и практических приемов построения программ для КО.

Программированное обучение возникло в 30-е гг. в США, в 60-е пережило ренессанс, на волнах которого получило широкое распространение во всем мире и достигло нашей страны. Оно представляет собой попытку повысить эффективность учебного процесса путем внесения в него элементов точности и научной строгости. Для этого взаимодействие между учащимся и обучающей инстанцией (книга или техническое устройство) разбивается на шаги, последовательность которых детально планируется — программируется — на основании данных психологии, педагогики и других наук, чтобы процесс усвоения знаний проходил с максимальной эффективностью и интенсивностью. Фактически программированное обучение является попыткой применить в области усвоения знаний идеи, предложенные в начале текущего столетия классиками научной организации труда Ф. У. Тейлором и Ф. Гильбретом, реализация которых привела к гигантскому росту производительности общественного труда.

Можно сказать, что характерный признак программированного обучения состоит в управлении усвоением знаний с помощью программы, детально определяющей ход обучающего диалога между учащимся и обучающим устройством, т. е. предъявление последним учебной информации, прием ответов учащегося и выполнение педагогических реакций на них. Очевидно, что под эту формулировку подпадает и любой обучающий диалог, осуществляемый с помощью компьютера. И здесь обязательной предпосылкой является наличие программы, обеспечивающей планомерное предъявление учебного материала и реагирование на ответы учащегося.

Заметим, что в литературе под программированным обучением часто понимается та его форма, которая сложилась в докомпьютерный период, когда программы оформлялись преимущественно в виде книг. Идея, как это нередко случается, была отождествлена с ее носителем. Но такое понимание неправомерно, КО и обучение с помощью программы в форме книги — все это лишь варианты, различные воплощения одной и той же идеи: программированного обучения. Этим вариантам присущи и различия. Для составления высококачественных обучающих программ для КО требуется определенный уровень знакомства с компьютером и знания его дидактических возможностей, так что приобретенный в прошлом опыт «книжного» варианта программированного обучения сам по себе недостаточен. Но он, безусловно, способен намного сократить требуемый для этого путь, подобно тому как опыт летчиков облегчает формирование профессии космонавта.

Характеристика программированного обучения была бы неполной, если не упомянуть о том, что сегодня его дидактические потенции иногда подвергаются сомнению. Но фактически при этом имеется в виду не идея программированного обучения, а лишь ее «книжный» вариант. Важнее, впрочем, другое. Как справедливо отмечает Н. Ф. Талызина [5], это разочарование обусловлено прежде всего тем, что качество программ было очень низким. Причин тому несколько, и главная среди них, вероятно, в том, что составлением программ занялись множество людей, часто не обладавших требуемой психолого-педагогической подготовкой.

Как свидетельствует опыт программированного обучения, исходная квалификационная база, позволяющая составлять достаточно эффективные программы, должна быть как минимум двойной: необходимо быть педагогом-предметником, т. е. иметь опыт преподавания того или иного учебного предмета, и психологом — специалистом по программированному обучению. Но, как правило, составителями программ были в лучшем случае только преподаватели-предметники, не владевшие психологическими знаниями. Их программы поэтому часто оказывались психологически несостоятельными. Отрицательно влияла на качество программ и неразработанность теоретических основ для их составления, вследствие чего работа велась, как правило, эмпирически, на уровне

интуиции [5]. Немалую роль в дискредитации программированного обучения сыграла также принципиальная ограниченность использовавшихся обучающих устройств, которые не позволяли реализовать достаточно совершенные программы. Не случайно главным средством обратной связи между учащимся и обучающим устройством стал метод выборочных ответов (после предъявления учебной информации дается вопрос и набор возможных ответов, из которых один или несколько правильные), отрицательные последствия которого для учащегося настолько значительны, что представляется оправданным стремиться по возможности к его исключению из практики школьного обучения [7].

Сегодня ситуация вокруг программированного обучения коренным образом изменилась. Компьютер позволяет уже сегодня, а тем более в перспективе воплотить в жизнь огромные потенции этой идеи и обеспечить в педагогическом процессе прогресс, сопоставимый с тем, что дали промышленному производству идеи научной организации труда. Улучшилось положение и в отношении теоретического осмысления путей реализации программированного обучения. Появились работы, создающие предпосылки для преодоления интуитивно-эмпирического подхода к составлению программ, для использования научно обоснованной модели учебного процесса [4], [6], [7].

Очевидно, что в этих условиях использование опыта психологов-специалистов по программированному обучению, являющихся к тому же преподавателями-предметниками, позволяет в максимальной степени ускорить решение проблемы разработки программ для КО.

Однако при создании в системе АПН, АН и бывшего Минпроса СССР упомянутого выше организационного механизма для подготовки программ этот опыт был, насколько нам известно, полностью игнорирован: ни один из названных специалистов-психологов к работе не был привлечен. Вместе с тем можно констатировать, что главной производительной силой в этом механизме стали программисты[2], которые, как правило, не владеют психолого-педагогической культурой, поэтому ожидать, что они за приемлемое время научатся составлять хорошие программы, нет оснований.

Но, может быть, в обучающих программах для КО главное — это их причастность к компьютеру и профессиональная компьютерно-математическая подготовка программистов позволяет компенсировать отсутствие у них квалификаций педагога-предметника и психолога-специалиста по программированному обучению? На этот вопрос необходимо ответить категорическим «нет».

В настоящее время имеется достаточно данных о работе программистов в качестве составителей программных средств для КО. Часто они создают инструментальные средства, т. е. программы, которые не являются обучающими и не могут использоваться самостоятельно. Например, программа, обеспечивающая преобразование предъявленных в символьной форме математических функций в наглядные графики, но не содержащая, полностью или частично, обучающего диалога. Полезная для учащегося информация здесь присутствует, но доступ к ней не создан. Инструментальные средства бывают весьма интересными, но для их использования нужна обучающая программа, которая включала бы их в качестве частей, к которым можно отсылать учащихся при постановке перед ними соответствующей учебной задачи. Без таких программ инструментальные средства остаются мертвым грузом.

Пытаются программисты создавать и обучающие программы, но они в психолого-педагогическом отношении часто оказываются несостоятельными, а порой и вредными для учащихся. Приведем пример. В программе обучающей правописанию букв З и С предъявляются слова типа «разделить» и «распахать» с пропущенной проблемной буквой, учащийся должен заполнить пробел. При правильном ответе на экране появляются рожицы — стилизованные изображения человеческого лица, числом от одной до четырех, после чего дается следующее задание. В случае ошибки появляется тележка, которая увозит неправильную букву и возвращается с правильной. Укажем некоторые недостатки программы. Есть основания думать, что она не столько обеспечивает, сколько тормозит формирование требуемого умения. Предъявление рожиц эквивалентно маске[3], стирающей воспринятую для решения задачи и подлежащую запоминанию информацию. Далее, недопусти-

[2] «Сейчас большинство компьютерных обучающих программ (КОП) создается на основе индивидуального опыта программистов, их чисто интуитивного представления о том, какой должна быть КОП» [3].

[3] Маска — используемый в психологических экспериментах бессмысленный стимул, предъявляемый после восприятия предыдущего стимула с целью стереть его след в памяти испытуемого, чтобы не препятствовать восприятию последующего стимула.

мо длительным является время ожидания (временной интервал между ответом учащегося и педагогической реакцией машины). Появление рожиц занимает около 8—13 с, столько же и движение тележки, и в течение этого времени возможность ввода информации исключается. Согласно психологическим данным, время ожидания не должно превышать 2 с. В данном случае оно больше нормы в 4—6 раз. Это значит, что через несколько минут работы с программой учащийся начнет скучать. Способствует появлению отрицательных эмоций также то, что после предъявления десяти заданий программа прекращает работу и для ее продолжения надо повторить всю процедуру ввода программы, занимающую продолжительное время.

Приведенные примеры весьма типичны. Программисты не в состоянии составлять высококачественные обучающие программы просто в силу отсутствия у них психолого-педагогической компетенции, вследствие чего единственной их опорой является интуиция. (Интересно отметить, что к сходным мыслям начинают приходить уже и сами программисты [1].) Весьма существенно в этой связи следующее обстоятельство. В настоящее время составление программ для КО осуществляется преимущественно в рамках частей учебных предметов — отдельных учебных тем или их фрагментов. Этот уровень имеет смысл рассматривать лишь как начальный, ориентировочный этап. Для развитой формы КО нормой должно стать программирование на уровне целостных учебных предметов, и единицами программного обеспечения должны стать компьютерные курсы. Только при этом условии можно ожидать, что возможности компьютера как инструмента повышения эффективности и интенсивности обучения будут использованы в полной мере. Но если программисты не в состоянии создавать обучающие программы на уровне учебных тем, то тем более бессильны они на уровне целостных учебных предметов.

Нам могут возразить. Проблема подготовки программ может быть решена на основе кооперации программистов с учителями, если в таком союзе незыблемым будет правило: «лидер, хозяин, заказчик — это педагог высокой квалификации» [1]. Однако это возражение при всей его правдоподобности, к сожалению, несостоятельно. Оно основано на допущении, что учителя *знают*, как составлять программы для КО, в действительности же они этого, как правило, *не знают*[4]. Ведь они незнакомы с психолого-педагогическими принципами программирования обучения, которым в институте не учат, а они имеют фундаментальное значение при разработке программ. Беспомощность учителей в этих вопросах отчетливо обнаружилась в 60—70-е гг. при попытках осуществления «книжного» варианта программированного обучения. Тем более нет оснований ожидать от них успеха (и практика это подтверждает), когда речь идет о несравнимо более сложном компьютерном варианте. Весьма существенно также, что большинство учителей незнакомы с компьютером и относятся к нему, если так можно выразиться, с суеверным почтением. Поэтому мнения программиста они склонны воспринимать как оракульские откровения, так что фактически сотрудничество сводится к диктату программиста. Этому способствует и то, что программисты подчас высокомерно и пренебрежительно относятся к задаче создания программ под руководством педагога [1] и к психологическим требованиям. Доминирование программистского опыта справедливо и для тех случаев, когда учителя владеют элементами компьютерной грамотности, обучившись на специальных курсах. Полученные там знания далеки от завершенности: овладеть за 15—40 ч уверенными программистскими умениями невозможно даже для представителей точных наук, не говоря уже об учителях-гуманитариях, так что и после окончания курсов программист остается для учителя неоспоримым авторитетом.

Подведем итог. Созданный в настоящее время в системе АПН и АН СССР для разработки обучающих компьютерных программ организационный механизм не в состоянии обеспечить решение этой проблемы, и ожидать, что ситуация самотеком достаточно быстро изменится в положительную сторону, нет оснований. Принципиальный дефект этого механизма состоит в том, что главной силой в нем является программист, не владеющий психолого-педагогической культурой и не способный поэтому обеспечить создание ни достаточно эффективных обучающих программ, ни теоретической основы, которая позволила бы ожидать появления таких программ в будущем. Попытка решить эту проблему путем объединения усилий программистов и учителей (методистов) в настоящее время бесперспективна в силу незнания ими принципов программирования

[4] Автор имел возможность лишний раз в этом убедиться при проведении с учителями школ Калининского района Москвы семинара «Подготовка сценариев для обучающих компьютерных программ».

обучения и компьютерной техники. В результате доминирующим при создании программ и определяющим их качество по-прежнему оказывается программистский опыт. Таким образом, высказанное в литературе предостережение о том, что «нельзя допускать захламления школы бессодержательными, отупляющими, хотя внешне и эффектными обучающими программами» [1], можно считать весьма актуальным.

Дефицит хороших программ и избыток плохих сегодня в известной степени маскируются вследствие необеспеченности школ компьютерной техникой. Но по мере поступления техники эти факторы будут все в большей степени становиться достоянием общественного сознания, что рано или поздно приведет к разочарованию, к дискредитации идеи КО. В свое время (60—70-е гг.) указанные факторы обусловили дискредитацию «книжного» варианта программированного обучения, но масштабы его распространения были незначительными, соответственно незначительным был и ущерб. Теперь же, когда речь идет о задаче, поставленной партией и правительством, область решения которой — вся страна, а на решение расходуются миллиарды, ущерб от дискредитации КО может быть огромным. Достаточно сказать, что нереализованным останется мощный потенциал компьютера как средства повышения эффективности и интенсивности учебного труда, труда преподавателей и учащихся.

Изложенные соображения позволяют утверждать, что организационный механизм, занимающийся в настоящее время проблемой подготовки программных средств для КО, нуждается в коренном изменении. Господствующий здесь интуитивно-программистский подход должен уступить место научному подходу. Для этого необходимо, чтобы при составлении программ доминирующим стал опыт, приобретенный при разработке проблематики программированного обучения, чтобы содержание и структура программ определялись носителями этого опыта, т. е. теми, кто в наибольшей степени обладает требуемой компетенцией. Это, как было показано выше, психологи-специалисты по программированному обучению, являющиеся вместе с тем педагогами-предметниками, некоторые из них владеют компьютерной техникой и опытом составления программ для КО. Разумеется, необходимо также участие методистов, программистов и специалистов других профессий, но интеграция совместных действий должна осуществляться психологами. Такая организация позволит перевести работы по подготовке программ на уровень высокого профессионализма, значительно ускорить ее и снизить затраты сил и средств.

Литература

1. *Вартапетян Э.* Не повторять старых ошибок // Информатика и образование, 1987. № 5. С. 114—115.

2. *Кулейников А. А., Супян В. Б.* Информационная техника: экономические и социальные аспекты // США: экономика, политика, идеология. 1987. № 3.

3. *Мирская А., Сергеева Т.* Обучающие программы оценивает практика // Информатика и образование. 1987. № 6. С. 49—53.

4. *Талызина Н. Ф.* Методика составления обучающих программ. М., 1980. 46 с.

5. *Талызина Н. Ф.* Компьютеризация и программированное обучение // Вопр. психол. 1987. № 6. С. 43—45.

6. *Шоломий К. М.* Построение обучающей программы // Информатика и образование. 1987. № 3. С. 52—58.

7. *Шоломий К. М.* Об одном подходе к использованию компьютера для обучения школьным предметам // Вопр. психол. 1987. № 5. С. 123—130.

8. *Carrier C. A., Sales G. C.* A taxonomy for the design of computer — based instructions // Educational technology. 1987. N 10. P. 15—17.

9. *Schiffman S. S.* Software infusion: using computers to enhance instructions // Educational technology. 1986. N 1. P. 7—11.

Б. ДРИНЬ, Р. СКУЛЬСКИЙ

Отбор содержания курса информатики

Введение в общеобразовательной школе курса ОИВТ поставило проблему отбора содержания образования по этой дисциплине, которая в настоящее время приобрела статус дискуссионной.

В публикациях неоднократно подвергались критике пробные учебные пособия «Основы информатики и вычислительной техники». К настоящему времени фактически нет таких учебных книг по рассматриваемому предмету, которые бы в полной мере соответствовали требованиям, предъявляемым к школьным

учебникам, что подтвердил, в частности, прошедший конкурс.

К основным причинам сложившейся ситуации можно отнести следующие.

1. Несоответствие школьного курса ОИВТ как учебного предмета структуре информатики как науки. Это можно объяснить тем, что к настоящему времени даже в кругу наиболее авторитетных представителей этой отрасли научных знаний не сложилось единого мнения о том, что является ее предметом и, соответственно, что должно войти в школьный учебник по данной дисциплине.

2. В существующем курсе информатики только частично реализованы многие дидактические принципы (научности, системности, последовательности в обучении и другие), вследствие чего учащиеся не имеют системных научных знаний по этому предмету.

Для устранения отмеченных негативных явлений, как минимум, необходимо решить следующие задачи:

осмыслить содержание школьного курса информатики в отношении научности его содержания;

обосновать критерии отбора основных понятий информатики как науки и возможность их включения в школьный курс;

подвергнуть содержание существующего школьного курса ОИВТ тщательному анализу для определения степени реализации в нем дидактических принципов обучения и воплощения прогрессивных педагогических идей.

Решение этих задач невозможно без глубокого знания информатики и вычислительной техники и теоретических основ содержания общего среднего образования.

В методическом пособии по информатике для учителей и преподавателей средних учебных заведений написано: «Общеобразовательное значение нового курса прежде всего связано с необходимостью существенного улучшения подготовки учащихся к практической деятельности в современном обществе...» Здесь же читаем: «Основная задача школы состоит в подготовке учащихся к использованию ЭВМ в последующей практической деятельности» [1]. Ориентируясь на перспективу, можно сформулировать данную задачу более широко — подготовка сегодняшних школьников к жизни в информационном обществе.

Разрабатывая данную проблему, авторы исходят из того, что на уровне учебного предмета одним из источников формирования содержания является наука, которая «стоит» за предметом. Следует помнить, что содержание не является сокращенной и упрощенной копией соответствующей науки, а представляет собой дидактически переработанную и обоснованную, предназначенную для целей общего образования систему знаний, умений и навыков, необходимых для применения в соответствующих видах деятельности. Другими словами, дидактическая модель учебного предмета — это некая целостность, включающая два блока: основной, куда входит в первую очередь то содержание, ради которого учебный предмет введен в учебный план, и блок средств, или процессуальный блок, обеспечивающий усвоение знаний, формирование различных умений, развитие и воспитание учащихся. Следовательно, учебник призван не только изложить материал, но и воплотить в себе методику работы учителя.

Анализ многочисленных публикаций в области школьной информатики свидетельствует о том, что в большинстве из них делается упор на основы алгоритмизации, программирования, вычислительной техники, численных методов и другое и очень слабо учитываются методологические, психолого-педагогические, психофизиологические основы компьютеризации образования.

Вследствие вышесказанного содержание существующего курса ОИВТ с его явно выраженной практической направленностью не может претендовать на статус общеобразовательного курса.

Какова же наша точка зрения по этому вопросу?

На февральском (1988 г.) Пленуме ЦК КПСС особое внимание было уделено проблеме дифференциации обучения. Ставится вопрос о создании специальных школ, классов с углубленным изучением отдельных дисциплин или их циклов, организации факультативных занятий для учащихся, проявляющих повышенный интерес к тому или иному предмету или практической деятельности, с таким расчетом, чтобы максимально удовлетворить индивидуальные интересы школьников.

В связи с этим наиболее целесообразной представляется идея дифференциации содержания образования по информатике и вычислительной технике. Оно не должно быть единым для всех школьников. Содержание, ориентированное на учащихся, не проявляющих особого интереса, должно нести в себе тот минимум знаний и основывающихся на них практических умений и навыков, которые нужны труженику любого профессионального профиля. Эта часть содержания школьного курса ОИВТ может быть определена следующими целями:

овладение учащимися знаниями об информации, ее видах, свойствах, методах и средствах сбора, хранения, преобразования; поня-

тие об ЭВМ как универсальном средстве работы с информацией;

овладение основами программирования; формирование у учащихся понятий об алгоритме, модели, исполнителе, программе, языке программирования;

приобретение учащимися навыков работы со своими программами и стандартным программным обеспечением.

При разработке содержания образования на этом уровне задача состоит в том, чтобы отобрать тот минимум знаний, который, будучи стабильным, политехнически ориентированным, был бы в то же время достаточным для дальнейшего пополнения знаний, для формирования современного научного стиля мышления, включал воспитательный и развивающий аспекты и не приводил к перегрузке учащихся [2].

Другая часть содержания образования (не обязательная для изучения всеми учащимися) должна включать развернутую систему знаний и практических умений, необходимых для специализации в одном из таких направлений, как, например, программирование или разработка и обслуживание ЭВМ.

Эта часть содержания образования, в свою очередь, должна быть рассчитана на различные уровня обучения (спецшколы, классы с углубленным изучением информатики и вычислительной техники, факультативы, кружки и т. п.) и строиться на основе принципа преемственности с уровнем предшествующей подготовки учащихся.

Следовательно, содержание образования по рассматриваемой дисциплине должно быть многоуровневым, например трехуровневым.

Первый уровень — минимальный, ориентированный на всех учащихся.

Второй уровень — на учащихся, проявляющих повышенный интерес к изучению информатики и вычислительной техники.

Третий уровень — для учащихся, проявляющих особые способности в данной области.

Разработка содержания образования по информатике на каждом уровне должна осуществляться с учетом реальных возможностей учащихся и их общеобразовательной подготовки по смежным дисциплинам (математике, физике и др.), материально-технической оснащенности школ, состояния и перспектив применения ЭВМ в различных отраслях народного хозяйства, науки и культуры.

Могут возникнуть вопросы о том, как быть тем учащимся, которые к началу изучения информатики не определили своего отношения к данному предмету и не имеют возможности заниматься на факультативах или в школах с углубленным изучением информатики; как сформировать у них интерес к данному предмету; как помочь тем, кто хочет знать немного больше, чем предусмотрено программой, интересуется лишь отдельными вопросами, но не имеет в достатке нужной литературы?

На наш взгляд, один из возможных выходов из таких ситуаций — это создание учебника, в котором был бы выделен обязательный материал (для усвоения всеми учащимися) и вспомогательный, который дал бы дополнительную, более обширную информацию для тех школьников, которые проявляют повышенный интерес к изучению информатики и вычислительной техники. Иначе говоря, речь идет о создании учебника, соответствующего по меньшей мере двум и даже трем разноуровневым учебным программам. Его практическое использование дало бы возможность реализовать принцип адаптации учащихся в учебном процессе в соответствии с их реальными возможностями, интересами и способностями. Кроме прочих дидактических преимуществ создание такого учебника было бы оправдано экономически, поскольку он в состоянии заменить 2—3 учебных пособия, в которых в значительной части повторяется один и тот же учебный материал.

Литература

1. *Ершов А. П., Монахов В. М.* и др. Изучение основ информатики и вычислительной техники. Ч. I. М.: Просвещение, 1985.

2. *Краевский В. В., Лернер И. Я.* Теоретические основы содержания общего среднего образования. М.: Педагогика, 1983.

В. ЛЯХОВИЧ

Практические занятия по информатике

В настоящее время, на наш взгляд, мало внимания уделяется содержанию практической части курса (занятиям, проводимым за дисплеем ЭВМ), методике проведения этих занятий.

Различается и содержание занятий: в одних школах много времени тратится на знакомство с клавиатурой, на изучение работы ЭВМ в режиме калькулятора, в других — почти половину практических занятий школьники изучают машинную графику и т. д. В одной школе, например, школьники тратили по десять с лишним часов на вычисление суммы членов ряда с точностью до 20-го знака!

Поэтому хотелось бы высказать свою точку зрения по этому вопросу. Обсуждая данную проблему, следует исходить из целей и задач практической части курса информатики, которые стоит сформулировать более четко.

Цели и задачи практической части курса ОИВТ

Цель — дать учащимся практические знания и навыки, которые обеспечат им возможность использовать ЭВМ в своей будущей работе; в средних специальных учебных заведениях эта часть курса должна обеспечить использование ЭВМ при изучении специальных предметов и возможность реализации компьютерной технологии обучения.

Задачами практической части курса следует считать следующие:

1. Дать учащимся представления и общие сведения по разделам:

состав операционной системы (ОС), структуры данных, пакеты прикладных программ;

этапы обработки программы в ЭВМ (ввод, трансляция, счет, вывод результатов);

содержательные и синтаксические ошибки в программе и диагностические сообщения ОС об ошибках.

2. Сформировать у учащихся навыки выполнения действий:

подготовка ЭВМ к работе;

решение на ЭВМ задач с использованием готовых программ;

отладка программ на ЭВМ;

организация библиотек на магнитных лентах и дисках для хранения данных и программ, работа с библиотекой;

редактирование текста и текстовых документов с помощью редактора текстов или других подобных средств;

получение твердых копий данных и программ; их тиражирование;

использование графических средств ЭВМ.

Средства достижения указанной цели

Сколько часов учебного времени нужно выделять для работы за дисплеем в курсе информатики? Обратимся к вузовскому опыту. В соответствии с учебной программой по программированию для технических вузов (первый уровень подготовки по вычислительной технике) на эти цели выделяется 20—30 ч. В то же время даже в ведущих вузах страны (например, МЭИ) время работы студента за терминалом ЭВМ в указанном выше курсе не превышает фактически 10—20 ч и при этом достигаются весьма удовлетворительные результаты. Поскольку требовать от школы более серьезной подготовки в области вычислительной техники, чем от вуза, вряд ли резонно, то, на наш взгляд, в школе достаточно выделять 16—18 ч на работу за дисплеем. К такому же выводу приводит и сопоставление сложности теоретического и практического разделов курса школьной информатики.

Практические занятия за терминалом ЭВМ следует проводить концентрированно, после обучения основам программирования, например только в X классе, в конце каждого полугодия. В противном случае чередование уроков с работой за дисплеем, малая частота практических занятий исключают возможность удовлетворительного изучения как теории, так и практической части.

Описание процесса выполнения каждого действия п. 2 должно излагаться в виде четкого алгоритма (см. рис.) и выдаваться учащемуся как раздаточный материал при выполнении им практической работы.

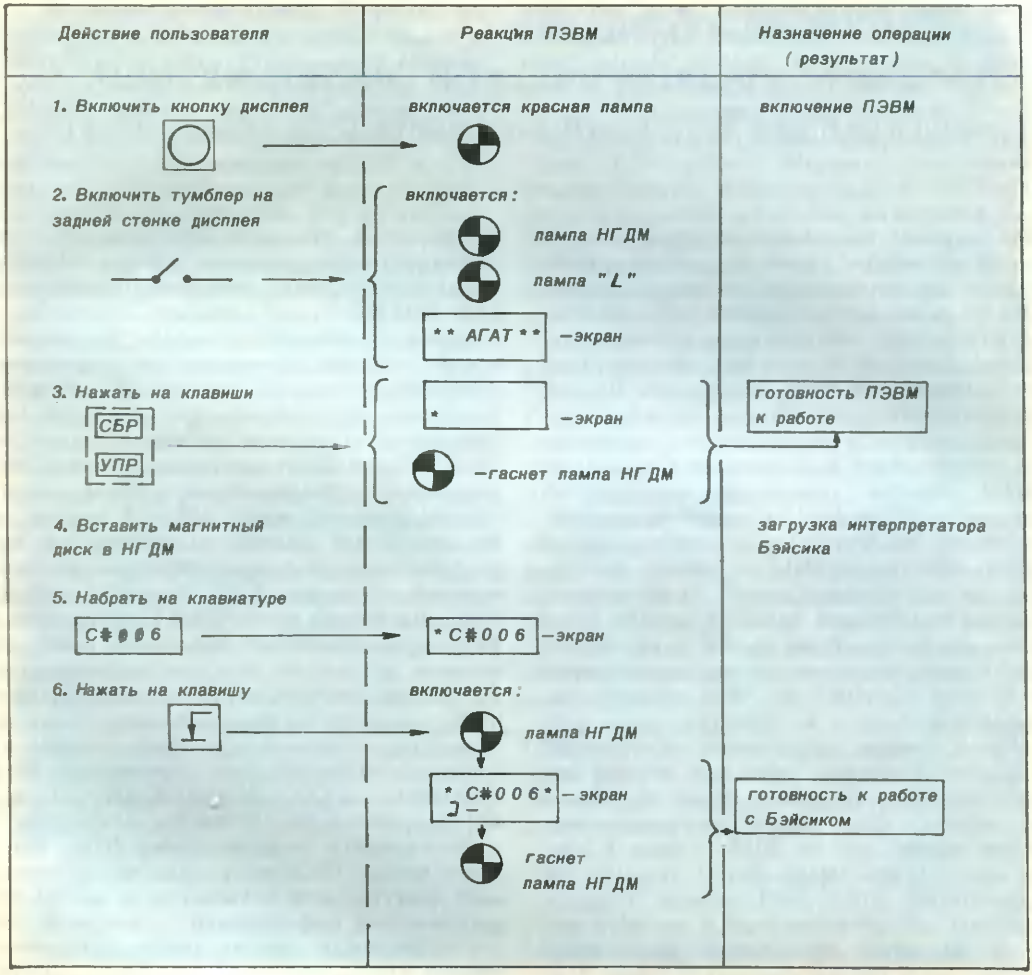
Изучение клавиатуры для экономии времени следует совмещать с изучением выполнения каких-либо работ за дисплеем, например с выполнением готовых программ.

Программа практических занятий (продолжительность каждого занятия 2 ч)

Занятие 1. Знакомство с ЭВМ, клавиатурой ЭВМ, выполнением готовых программ:

подготовка ЭВМ к работе;

вызов на экран готовой программы с магнитного диска (МД), ленты (МЛ);



простейшее редактирование программы (например, изменение имени программы, арифметического выражения);

запуск программы на счет.

Занятие 2. Знакомство с сообщениями ЭВМ о типичных синтаксических и содержательных ошибках:

вызов на экран программы с МД (МЛ);

умышленное искажение текста;

запуск программы на счет;

изучение сообщения об ошибке.

Занятие 3. Отладка простейшей (линейной) программы:

ввод программы с клавиатуры дисплея;

отладка программы;

запись программы на МД (МЛ).

Занятие 4. Отладка разветвляющейся программы с линейными таблицами.

Занятие 5. Отладка циклической программы (с подпрограммами).

Занятие 6. Отладка программы с несколькими циклами.

Занятие 7. Обработка текстовой информации:

ввод текстового документа с МД (МЛ);

редактирование текста;

запись документа на МД (МЛ);

вывод документа на печать.

Занятие 8. Использование графических средств ЭВМ:

вызов на экран с МД (МЛ) готовой программы изображения графика функции; запуск программы на счет;

редактирование программы и повторный запуск на счет;

ввод с клавиатуры программы изображения чертежа, например, простейшей детали; ее отладка и запуск на счет; запись программы на МД (МЛ).

Занятие 9. Контрольная работа.

В. ГУЗЕЕВ
МОПИ им. Н. К. Крупской

О технологии обучения и предмете педагогической информатики

(Размышления над концепцией информатизации образования)*

Предложенная концепция на первый взгляд четкая и ясная, хотя и многословная; разработаны этапы информатизации, поставлены задачи методического обеспечения, честно сказано о трудностях и уже допущенных ошибках. Однако не оставляет ощущение неполноты этих построений. Думаю, слабость концепции — в том, что в ее основу положено понятие «новая информационная технология». В педагогическом смысле НИТО должна содержать совокупность приемов, методов, форм обучения на компьютерной основе. Может ли кто-нибудь сейчас сказать, что такое НИТ в школе, если не произносить общие фразы? Традиционная технология обучения известна, можно говорить о лекционно-семинарской и некоторых других технологиях, а вот информационной технологии обучения нет. Как внедрять то, чего нет?

Много сказано о развитии технической базы. Нет сомнений, рано или поздно она будет создана (и даже вряд ли можно предсказать, какие еще технические новшества удивят нас до 2010 г. Беда в том, что мы получаем мощнейшую технику, не зная толком, что с ней делать. Тогда и начинаем поворачивать реки и осушать моря! К развитию технической базы школ должны быть готовы соответствующие технологии обучения. Более того, техническая сторона должна быть подчинена педагогической: создание и развитие технологий обучения должно предвосхищать техническое оснащение школ. Иначе мы многократно повторим печальный опыт внедрения ТСО, которые и по сей день во многих случаях не используются.

В концепции много слов о методическом и педагогическом обеспечении информатизации. Но что первично? Одна из задач методики (и педагогики) — критерии выбора адекватной технологии. Правильно говорят авторы о рынке педагогических технологий. А раз так, то надо вести речь о компьютерной поддержке методик, а не наоборот. Указывается, что ныне практически отсутствует методический задел работы с ПЭВМ. На деле отсутствуют методики, нуждающиеся в ПЭВМ!

Наверное, настало время для оформления новой научной дисциплины — педагогической информатики. Создание НИТ обучения и должно быть одной из целей ее. Обсуждаемая концепция должна быть дополнена тогда концепцией педагогической информатики. Предложенные этапы можно сделать более четкими. На мой взгляд, на базовом этапе задача педагогической информатики — разработка интегральных технологий обучения, т. е. развитие лучших некомпьютерных технологий с включением в них информационных элементов, обеспечивающих достижение каждым учеником хотя бы минимального, репродуктивного уровня, чтобы создать учителю плацдарм для развивающего обучения. Это требует создания систем пакетов программ-тренажеров по всем предметам (АУК МИНИМУМ). В это время в школе накапливается опыт работы с имеющимися разрозненными ППС, готовятся кадры. На первом этапе школа осваивает интегральные технологии, а задача педагогической информатики — создание инструментальных средств реконструктивного уровня (моделирующие программы, имитационные среды, программы-исполнители и т. д.), включаемых в интегральные технологии второго уровня. На втором этапе в школу внедряются эти варианты технологий, а очередная цель НИР — разработка технологий креативного уровня (компьютерные лаборатории, системы телекоммуникаций, экспертные системы и т. д., о чем много сказано в концепции). На перспективном этапе — методическая и психологическая разработка педагогических систем искусственного интеллекта и пересмотр самых основ школьного образования. Впрочем, верно сказано, что детальный прогноз невозможен. Хочу лишь подчеркнуть, что исследования по педагогической информатике должны значительно опережать массовую поставку в школы технических средств.

Сейчас нередко программисты готовят ППС, выбирая то, что легче программиро-

* Концепцию информатизации образования см.: Информатика и образование. 1988. № 6.

вать, а затем ищут методистов, чтобы сделать методическое обеспечение: вот программа, куда бы ее приспособить? Я несколько преувеличиваю, конечно. Но суть в том, что должно быть наоборот. Специалисты готовят технологию обучения и, исходя из нее, прорабатывают сценарии ППС. Эти сценарии передаются коллективам программистов. Изготовленные ими программные продукты возвращаются на методическую (психологическую, медико-гигиеническую и т. д.) экспертизу, экспериментально проверяются, дорабатываются и затем тиражируются, включаясь в уже готовую технологию обучения. Разработчиком сценария может быть и учитель-практик, и преподаватель вуза, и коллектив учащихся, и кто угодно, но важно, чтобы созданный сценарий был перед программированием оценен специалистами. Для этого мы должны иметь орган, в котором печатались бы сценарии ППС, или канал, по которому эти разработки практиков попадали бы к экспертам-методистам. Кстати, этим же каналом могли бы пользоваться и называемые в концепции «точки опережающего роста», чтобы заявить о себе. Может быть, журналу «Информатика и образование» организовать специальный раздел или приложение?

Из обсуждаемой концепции практически выпали педвузы. А напрасно! Исследования по педагогической информатике лучше всего сосредоточить там, где ведется подготовка педагогических кадров, ведь их обучение тоже должно носить опережающий характер. На базовом этапе целесообразно открыть проблемные лаборатории при методических кафедрах вузов, располагающих квалифицированными преподавателями, с небольшим собственным штатом и широким привлечением студентов к НИР. На первом этапе эти лаборатории могут вырасти в кафедры педагогической информатики, на некоторых факультетах откроются отделения педагогической информатики и АСУНО. На втором этапе возможно разделение кафедр на более специальные и открытие факультетов педагогической информатики и информационно-управляющих систем народного образования. На перспективном этапе неизбежен пересмотр всей структуры факультетов в педвузах. Все эти подразделения на всех этапах должны иметь доступ к информации о любых отечественных и зарубежных исследованиях, свободу научного обмена, приоритетное финансирование и техническое оснащение.

И в заключение об одной частности в обсуждаемой концепции. Термин «компьютерная педагогика» представляется столь же нелепым, сколь и «компьютерное воспитание»,— педагогика не может быть прерогативой каких бы то ни было технических систем: человека воспитывает только Человек!

Всё мною сказанное не относится к курсу ОИВТ — о нем и так много говорят. Этот курс — деталь в механизме информатизации, имеющая гипертрофированное значение лишь на первом этапе.

В. КАЙМИН

К концепции информатизации образования в СССР

Стартом для обсуждения проблем информатизации образования в СССР для нас послужила статья академика А. П. Ершова «От компьютерной грамотности учащихся к информационной культуре общества», опубликованная во 2-м номере журнала «Коммунист» за 1988 г. Главным в ней, на наш взгляд, является тезис о переходе общества, прошедшего путь от аграрного к индустриальному, к новому уровню развития: информационному обществу. Этот новый уровень связывается, с одной стороны, с использованием средств информационной и вычислительной техники во всех сферах жизни — в производстве, проектировании, науке, управлении, образовании, здравоохранении, быту и т. д.; с другой стороны — с новым уровнем развития культуры людей, называемой информационной культурой.

Для обсуждения любых проблем первостепенное значение, как известно, имеют определения основных понятий, среди которых мы хотели бы выделить следующие: 1) информационные технологии; 2) компьютерная грамотность; 3) информационная культура. Ключом к определению этих понятий является слово «информация». В самом простейшем толковании информация —

это сведения о ком-то или о чем-то, которые могут передаваться в устной, письменной форме или с помощью каких-то технических средств.

Исходя из этого легко дать толкование понятию «информационные технологии». Это — передача, накопление и переработка информации с помощью технических средств — карандашей и бумаги, холста и кисти, пишущих машинок и типографских машин, телефона и телеграфа, радио и телевидения, магнитных кассет и грампластинок и, конечно же, с помощью современных компьютеров и средств вычислительной техники, которые на наших глазах преобразуют технологии производства материальных и духовных ценностей.

Отсюда вытекает необходимость определения понятия «компьютерная грамотность» и способов ее достижения всеми — и учащимися, и взрослым населением. Ключом к пониманию словообразования «компьютерная грамотность» является слово «грамотность». На самых ранних этапах образования она означает умения читать, писать, считать, рисовать, работать с книгами и тетрадями. Теперь понятно, что «компьютерная грамотность» — это умение работать с информацией на ЭВМ: читать и писать, считать и рисовать, искать и накапливать информацию и работать с компьютерными программами. По нашим наблюдениям этим умениям дети в возрасте от 10 до 15 лет обучаются за 16—18 часов, а взрослые — за 10—12 часов.

Овладение компьютером как средством подготовки, накопления и обработки информации самых различных видов открывает путь к приобретению информационной культуры. И теперь понятно, что такое информационная культура. В самом широком смысле — это умение передавать информацию людям в устной, письменной, графической или иной форме с помощью бумажных или других средств, в том числе с помощью компьютеров и линий связи. По отношению к компьютерам информационная культура означает умение накапливать, искать и передавать информацию, а также решать свои задачи с помощью ЭВМ. Формированию этих умений служит курс «Основы информатики и вычислительной техники», введенный в Советском Союзе в среднее образование с 1985 г.

Решение поставленных проблем требует серьезных фундаментальных исследований, прикладных разработок и организации крупномасштабных экспериментов в системе народного образования. К числу фундаментальных исследований мы относим работы в области:

1) новых технических средств обучения, включающих информационно-вычислительную технику;

2) новых методов обучения на базе перспективных и существующих информационных технологий;

3) психолого-педагогических аспектов обучения в рамках новых информационных технологий;

4) социально-экономических аспектов использования средств вычислительной техники во всех сферах жизни.

Важнейшие прикладные исследования, необходимые для широкого применения ЭВМ в системе народного образования, на наш взгляд, включают разработку:

1) баз знаний и экспертных систем по всем учебным дисциплинам;

2) компьютерных курсов как основы для лабораторных практикумов на базе ЭВМ;

3) моделирующих программ как средств демонстрации различных процессов на базе ЭВМ и видеотехники;

4) обучающих тренажеров в качестве средств индивидуализации обучения;

5) всевозможных электронных конструкторов как средств обучения проектированию с помощью ЭВМ;

6) новых инструментальных средств для разработки новых учебных компьютерных программ;

7) различных баз данных для управления процессом обучения и подготовки информации для руководства.

К первоочередным задачам, на наш взгляд, следует отнести прежде всего:

1) обеспечение всех школ и учреждений среднего образования надежной учебной вычислительной техникой;

2) разработку качественного программного обеспечения для существующих учебников по информатике;

3) подготовку полного учебно-методического обеспечения для учителей информатики;

4) организацию углубленной подготовки учителей информатики и создание учебно-методических центров компьютерного обучения для всех учителей и руководящих работников системы образования.

Зарубежный опыт

Как торгуют компьютерами в ГДР?

На проходившей в конце прошлого года на ВДНХ СССР выставке «ГДР в Москве» немало экспонатов так или иначе были связаны с вычислительной техникой и программой компьютеризации, которая ныне осуществляется в ГДР. Это персональные компьютеры и электронные пишущие машинки фирмы «Роботрон» и система программирования 8- и 16-битовых микропроцессоров, выпуск которой налажен на берлинском комбинате электроаппаратов «Фридрих Эберт». И все же наибольшее впечатление произвел на нас стенд комбината микроэлектроники «Вильгельм Пик» (г. Мюльхаузен). Здесь разместилась микро-ЭВМ КС85/4, которая пришла на смену предыдущей модели КС85/3, использовавшейся в ГДР в системе профессионально-технического обучения. Прежде чем перечислять ее основные технические характеристики, стоит напомнить, что совсем еще недавно, когда в московском магазине «Электроника» уже появились первые экземпляры БК-0010, в ГДР слова «домашний компьютер» ассоциировались разве что с зарубежной покупкой. Прошло совсем немного времени, и сегодня в ГДР любой желающий без какой-либо предварительной записи да, впрочем, и без очереди может приобрести компьютер. Соответствующие отделы открылись в радиомагазинах всех окружных центров и почти во всех крупных городах страны. И это при том, что компьютеры пользуются спросом. Динамичность и умение точно планировать спрос в общем-то неудивительны, когда речь идет о промышленности ГДР. Интересно разобраться, чем объясняется такой спрос на довольно дорогое изделие (стоит компьютер примерно столько же, сколько БК-0010.01). Инженер предприятия микроэлектроники Карстин Шинон пояснил нам причины популярности своего компьютера.

Прежде всего это, конечно, характеристики машины: 8-разрядный микропроцессор 880, оперативное запоминающее устройство 128К байт, причем 64К байт памяти отдано в распоряжение пользователя, а 64К байт занимает память экрана. Такой объем памяти, отданной под экран,— признак богатых графических возможностей, — на экране 320×256 элементов можно выбирать цвет из 8-цветовой палитры, причем 64К байт экранной памяти в свою очередь разделены между двумя страницами экрана. Каждый из встроенных в компьютер генераторов, а их два, позволяет воспроизво-

Домашний «центр информатики» в комплекте: телевизионный приемник — дисплей, магнитофон — накопитель информации на магнитной ленте, отделяющаяся клавиатура, микро-ЭВМ КС 85/4, дополнительные блоки.

дить пять октав, программно регулируется громкость звука (32 уровня). Звук смешивается с видеосигналом, поэтому усилителем служит телевизор, к которому подключен компьютер. О подключении к телевизору — этом «узком месте» нашей «Электроники БК» — стоит поговорить подробнее. У компьютера предусмотрено несколько выходов — антенный позволяет без каких-либо переделок или доработок подключить компьютер к любому телевизору производства ГДР, впрочем, как и к любому другому телевизору цветного или черно-белого изображения. Лучшего качества можно добиться, соединив видеовыход компьютера с соответствующим входом (нельзя не отметить: в телевизорах с маркой RVT этот вход имеется, поэтому «хирургическое вмешательство» для подключения компьютера не требуется). Еще одна приятная деталь — клавиатура выполнена отдельным небольшим блоком. Это 64 клавиши в стандартном расположении, «дребезг» практически отсутствует. Функции шести клавиш-ключей программируются. В качестве внешней памяти используют обычный кассетный или катушечный магнитофон. Характеристики неплохие, однако особенно привлекательным показалось нам другое. Поразил комплексный подход к производству и торговле бытовыми компьютерами, который принят в ГДР.

Проиллюстрировать это можно на примере того же комбината. Здесь считают, что сделать основной блок компьютера — это лишь часть работы. Одновременно с микро-ЭВМ был запущен в производство и появился в магазинах целый набор дополнительных устройств. Вот их неполный перечень: дополнительное ОЗУ 16К байт, дополнительное ОЗУ 64К байт, перепрограммируемое ЗУ пользователя 8К байт, аналогово-цифровой преобразователь, цифро-аналоговый преобразователь, модуль сопряжения с устройствами пользователя, аналог последовательного интерфейса РС232, аналог параллельного интерфейса «Центроникс» и др. Особо отметим лишь блок ПЗУ с ассемблером, дизассемблером и сервисными программами, блок ПЗУ с языком Форт, разрабатывается модуль для подключения дисковода. Так на деле находят подтверждение обещания, что компьютер можно использовать для малой автоматизации, для сбора и статистической обработки результатов экспериментов. Компьютер приспособлен и для редактирования текстов. Отредактированный на компьютере текст — пустая забава, до тех пор пока этот текст нельзя отпечатать на бумаге — вывести на принтер, как говорят пользователи. Где взять принтер? Этот вопрос, пожалуй, больше всего волнует владельцев БК-0010. У владельцев КС85/4 проблем нет — в обычном универмаге продается электронная пишущая машинка (например, «Эрика 3004» фирмы «Роботрон»). Помимо своего основного на-

Несмотря на свои скромные размеры, эта настольная 16-разрядная мини-ЭВМ способна работать в режиме разделения времени одновременно с 8 терминалами. В этом случае общая память, доступная каждому пользователю, составляет 40 Мбайт, однако ее можно увеличить вдвое, подключив дополнительный блок. Если машина используется для управления технологическими процессами, то к ней подключают блок, обеспечивающий работу в режиме реального времени. Выпущено уже более 2000 таких машин, а спрос на них не уменьшается. Среди прочих достоинств то, что мини-ЭВМ работает под управлением операционной системы «Вега», практически полностью совместимой с известной ОС «UNIX».

С помощью этих блоков типовую микро-ЭВМ КС 85/4 можно настроить на решение ваших задач. Уже сегодня есть блоки от расширения ОЗУ и ПЗУ до подключения принтера, от ЦАП и АЦП до модулей с языками программирования или другими, часто употребляемыми программами.

значения машинка способна работать в качестве принтера, стоит лишь запастись переходным блоком. Нельзя не отметить, что распечатки получаются высококачественными, выгодно отличаясь от тех, что выдают мозаичные принтеры. Разве что отсутствие опечаток и исправлений указывает на то, что текст готовился с помощью компьютера. «Железо» — как нередко называют компьютер, принтер и другие периферийные устройства — это лишь половина дела. Второй из двух китов, которые составляют основу понятия «домашний компьютер»,— это программное обеспечение. Вопросу «софта» — программного обеспечения — на комбинате уделяют, пожалуй не меньшее внимание, чем схемотехническим разработкам. Упор, однако, делается на системные программы, прежде всего на операционную систему, которая поставляется вместе с компьютером, и на языки программирования. Тем не менее сегодня в том же отделе, где торгуют компьютерами, можно приобрести игровые и учебные программы, специальные программы. Как они появились? Причина тому — активность пользователей и усилия, которые предпринимают на комбинате, чтобы увеличить тиражируемый программный фонд. В ход идут различные средства: программы заказывают в организациях, где используются эти микро-ЭВМ, немало программ составляют владельцы компьютеров — некоторые из них комбинат покупает, многие получает в подарок. Каждые полгода издается список тех программ, о существовании которых удалось узнать, хотя не все из этих программ можно приобрести в магазине. Это координирует усилия пользователей, каждый владелец компьютера старается создать что-то новое, не тратя времени на повторение этапов уже пройденных другими.

Разумеется, это не полная информация о компьютерах ГДР или деятельности предприятия микроэлектроники «Вильгельм Пик». Это лишь хороший пример, следовать которому могли бы и наши предприятия, занимающиеся выпуском домашних компьютеров, будь то БК или «Микроша», «ПК-01 Львов» или «Криста», «Апогей» или «Немига». А может быть, за дело возьмутся кооперативы?

А. БОЙКО

ВНИМАНИЮ ЧИТАТЕЛЕЙ!

В декабре 1988 г. в Московском центре международной торговли состоялась Учредительная конференция Международного компьютерного клуба. Учредителями клуба стали В/О «Знание», АН СССР, ВЦСПС, а также ведущие банки, институты и вузы страны. Членами клуба могут быть советские и иностранные фирмы, организации, а также частные лица.
Подробную информацию о клубе читайте в нашем следующем номере.

Адрес клуба: 101813, Москва, Центр, проезд Серова, 4
Телефон: 921-09-02
Телекс: 411630

In December 1988 Moscow's International Trade Centre hosted a foundation conference staged to establish a non-governmental public organization, International Computer Club, its founders being All-Union Society "Znanie", USSR Academy of Sciences, All-Union Central Council of Trade Unions, as well as the country's major banks, research and higher education institutions. Both Soviet and foreign firms, organizations and individuals are welcome to become members of the club. Further information on the club you will find in our next issue.

Address: International Computer Club, 4, Proyezd Serova, Centre, Moscow, 101813, USSR
Phone: 921-09-02
Telex: 411630

Тема семинара — компьютеризация школы

С 1987/88 учебного года Московский городской институт усовершенствования учителей проводит научно-методический семинар «Современная концепция компьютерного урока», посвященный широкому кругу проблем компьютеризации школы.

Необходимость такого семинара подсказала сама жизнь. Обозначились проблемы дефицита общения творчески работающих учителей, а также нехватки информации об исследованиях ученых-методистов и психологов, о работе наших коллег за рубежом, о различных событиях, происходящих в сфере школьной информатики[1].

Компьютеризация школы, начавшись с введения в старших классах нового общеобразовательного предмета «Основы информатики и вычислительной техники», отнюдь им не исчерпывается. Не ограничиваются им и задачи компьютеризации школы. Здесь и сдвиг курса в средние и даже младшие параллели, и применение компьютеров не как объекта изучения, а как универсального средства для обработки информации, обучения другим предметам, для административных нужд, в том числе и школьного самоуправления, для организации общественно полезного труда и профессиональной подготовки, использование компьютеров для внеклассной работы и т. д.

Единственным человеком, кто может и начинает решать эти задачи, оказывается учитель информатики. Так, кстати, происходит во всем мире.

Однако наш учитель попадает в довольно сложную ситуацию, ведь и основная деятельность — обучение информатике — дает много поводов для головной боли: начиная от неустоявшегося содержания и почти неразработанной методики преподавания и продолжая (но не заканчивая) отсутствием книг, пособий, программного обеспечения, не говоря уже о постоянной заботе об исправности техники.

Работать тем не менее надо, и каждый начинает как-то выходить из положения. Как результат, с одной стороны, разнообразие идей и их воплощений, а с другой — плодятся изобретения велосипедов, множатся дублирующие друг друга программные поделки, распыляется большой интеллектуальный потенциал[2].

Таким образом, наш семинар был призван хоть немного снять остроту дефицита информации, дать пищу для размышлений его участникам, толчок в их творческой деятельности, а также помочь им завязать между собой контакты для возможного коллективного творчества (так в ряде случаев и получилось, например сформировалась методическая комиссия, выработавшая содержание московской олимпиады по информатике, сложился коллектив, написавший задачник по информатике (см.: Информатика и образование. 1988. № 4), есть и другие примеры).

Заседания семинара проходили ежемесячно в тех школах и УПК, где можно было познакомиться с интересным опытом работы. Присутствовавшие слушали преподавателей этих школ и УПК, а также приглашенных ученых, методистов, разработчиков программного обеспечения и активно обсуждали то, что было предложено их вниманию.

Чтобы дать более полное представление о тематике семинара, упомянем некоторые выступления.

Так, во время заседания, проходившего в школе № 209 Свердловского района, учитель В. А. Урнов рассказал о своей методике индивидуальной работы с учащимися, о проекте компьютеризации школы в условиях ограниченного количества техники (сейчас он претворяется в жизнь), этот опыт будет представлен на ВДНХ СССР. Учитель Д. Ю. Климов познакомил собравшихся с учебным пособием по информатике для пользователей КУВТ-86, -87 и ДВК, которое подготовлено им в соавторстве с В. А. Урно-

[1] К сожалению, недостаток оперативной информации — был, есть и остается, несмотря даже на существование журнала «Информатика и образование»: как правило, здесь публикуются (если публикуются вообще) сообщения о событиях давно прошедшего времени. Читая о них, можно только приобрести комплекс неполноценности на почве безнадежной отсталости. Когда же мы будем вовремя узнавать о конференциях и семинарах, летних школах и конкурсах, выставках и многом другом? Создается впечатление, что те, кто имеет доступ ко всей этой информации, как у М. М. Жванецкого, «сами ее и потребляют».

[2] Автору статьи близка мысль, высказанная В. А. Кайминым о том, что по количеству научных работников, программистов и инженеров, пришедших сейчас работать в школу, коллектив московских учителей информатики вполне можно рассматривать как распределенный научно-исследовательский институт.

вым (книга выйдет в издательстве «Просвещение» в 1989 г.).

В УПЦ Октябрьского района профессор А. Л. Брудно рассказал об используемой здесь методике профессиональной подготовки учащихся в области ЭВМ (в УПЦ занялись этими вопросами едва ли не раньше всех в Союзе), в частности о профиле «программист-лаборант», а также о многолетнем опыте проведения московских олимпиад по программированию (олимпиадные задачи частично опубликованы в книге А. Л. Брудно и Л. И. Каплана «Олимпиады по программированию», вышедшей в 1985 г. в издательстве «Наука»).

В спецшколе № 45 Севастопольского района учитель М. Э. Кушнир познакомил участников семинара с опытом организации оплаченного производительного труда старшеклассников в области ЭВМ (об этом писала в феврале 1988 г. «Литературная газета»), а также со своей разработкой — инструментальным педагогическим программным средством — электронным задачником. Участники семинара получили огромное удовольствие от посещения занятия кружка по программированию на языке Лого для учеников I—III классов, который ведет преподаватель В. М. Легоньков.

В школе № 167 Ленинградского района учитель А. И. Ковнер продемонстрировал в действии свою разработку для СМ-1420 — программное средство, которое позволяет писать и исполнять программы на языке, практически совпадающем (добавлен только ввод/вывод) с алгоритмическим языком учебного пособия (статью А. И. Ковнера на эту тему см.: Информатика и образование. 1987. № 5).

В школе № 314 Куйбышевского района учитель З. М. Штильман рассказала о своем опыте обучения формальной записи на Фокале алгоритмов, разработанных на алгоритмическом языке. (Статья З. М. Штильман в соавторстве с Б. М. Штильманом, в которой предлагалась эта методика, была опубликована в журнале «Микропроцессорные средства и системы», 1986, 3.)

Кроме того, на заседаниях семинара с докладами, посвященными концепциям, положенным в основу перспективных учебных пособий, их содержанию и структуре, выступали представители авторских коллективов В. А. Каймин и А. Г. Кушниренко.

Представители двух других авторских коллективов — Б. М. Штильман и В. Ю. Милитарев познакомили участников семинара с оригинальными проектами программ факультативного курса по информатике (см.: Информатика и образование. 1988. № 1) и курса основ информатики для физматшкол (последний проект представляет собой скорее альтернативный подход к курсу информатики вообще, его обсуждение вызвало большой интерес присутствующих).

А. В. Гиглавый рассказал о перспективных моделях школьных компьютеров за рубежом и в СССР.

Практическую разработку мощного инструментального средства для поддержки компьютерного урока представил А. Е. Новодворский — один из членов авторского коллектива программистов-разработчиков под руководством А. А. Фокина.

Методист МГИУУ Е. С. Крылова рассказала об опыте компьютеризации школ г. Софии (НРБ).

Д. Ю. Климов познакомил собравшихся с исследованиями в области разработки аппаратных средств искусственного интеллекта. Были сделаны и другие интересные сообщения и доклады.

Члены семинара выразили удовлетворение его работой в течение года. Было решено, что семинар продолжит свое существование. Заседания проходят по третьим четвергам каждого месяца в помещении МГИУУ, начало в 16.00.

Е. КРЫЛОВА

Еще раз о здоровье...

Предмет «Основы информатики и вычислительной техники» был введен в школах без глубокого предварительного анализа социальных, психологических и других последствий этого шага. Одним из важнейших вопросов, которые не были исследованы, является влияние работы с компьютером на состояние физического и психического здоровья участвующих в учебном процессе. Новый предмет был введен без согласования с Министерством здравоохранения! А между тем сведения, которые приводятся специалистами, более чем настораживают.

Многочисленными исследованиями установлено, что параметры ряда наших ПЭВМ не удовлетворяют гигиеническим нормативам. Для детей высока опасность развития фотоэпилепсии, ухудшения зрения, появления неврологических симптомов. Болезням, связанным с компьютеризацией, посвящают-

ся международные конгрессы. И, по-видимому, недаром 16 стран в 1985 г. приняли норму работы операторов за пультом ЭВМ менее 8 ч в день [1, 2].

Нельзя сказать, что ничего не делается для снижения отрицательного влияния компьютеров на здоровье школьников. Предпринимаются попытки освоить ПЭВМ с мониторами, более или менее отвечающими международным стандартам. Введены «Временные рекомендации по режимам занятий учащихся за видеотерминалами ЭВМ» [1]. В печати изредка появляются сообщения об исследованиях влияния работы с компьютером на состояние здоровья детей [3], [4].

Но хороших ПЭВМ пока нет, когда они будут — неизвестно [5]; время работы школьников с ЭВМ в реальных условиях школы ограничить очень трудно. И если что-то для обследования и охраны здоровья школьников еще делается, то практически исключены из поля зрения учителя, студенты, преподаватели вузов и обслуживающий персонал ЭВМ.

Между тем, мне кажется, из всех перечисленных категорий наиболее критическим является положение преподавателя информатики в педвузе. Во-первых, его нагрузка составляет 900—1000 ч в год, и у ассистентов эти часы представляют собой в основном аудиторную нагрузку, т. е. такую, которая связана с работой в дисплейном классе. Поделим общую сумму на количество недель занятий в двух семестрах: 10 недель в первом (сокращение за счет сельскохозяйственных работ) плюс 16 недель во втором — итого 26 недель. Получаем 34,6—38,5 ч в неделю. Если учесть еще неравномерность распределения занятий в семестре, будет понятно, почему я, например, апрель 1988 г. работала в дисплейном классе по 8—12 ч в день. Во-вторых, необходимо прибавить еще психологическую нагрузку, которая, как показывает практика [6], [7] и подтверждают теоретические исследования [8] существенна и обусловлена следующими особенностями работы преподавателя информатики.

формирование умения составлять алгоритмы программ возможно лишь при проблемном обучении, что предполагает увеличение разнообразия и повышение уровня сложности задач, решаемых в классе одновременно;

необходимо оперативное управление одновременной исследовательской активностью всех учеников класса в отличие от традиционных условий обучения, когда оперативное педагогическое воздействие обычно носит индивидуальный характер, а оценка и коррекция работы учащихся являются отсроченными;

характер ошибок, которые ставят обучаемых в тупик и требуют вмешательства преподавателя, может быть самым различным — от неправильного принципа решения до неверно нажатой клавиши.

В этих условиях преподаватель должен достаточно быстро оценить ситуацию, в которой оказывается тот или иной ученик или студент, понять характер затруднения и оказать необходимую помощь, что требует значительных усилий.

На мой взгляд, нельзя и дальше закрывать глаза на проблему охраны труда преподавателей.

В одной из работ [8] психолог отмечает, что преподавание информатики в школе представляет собой, по существу, эксперимент по формированию психических качеств человека. Только необходимо отметить, что эксперимент не ограничен названным направлением. Главное — эксперимент идет над здоровьем наших детей, больших, и маленьких, и тех, которых еще вынашивают беременные студентки и преподавательницы.

Мне кажется, объективный анализ будет выполнен, если развернуть массовые обследования учащихся и педагогов во время учебного процесса в школах и в вузах, проводить их силами компетентных специалистов. Кроме того, по-видимому, необходимо изменить систему планирования нагрузки вообще.

«Результаты применения ЭВТ необходимо соотносить с социально-ценностными ориентациями и целями развития всей общественной системы социализма, преимущества которой не реализуются автоматически» [9]. Эти слова Г. Смоляна подчеркивают важность данной проблемы.

Литература

1. *Маслов А., Таиров О., Труш В.* Физиолого-гигиенические аспекты использования персональных ЭВМ в учебном процессе // ИНФО. 1987. № 4. С. 79—80, 91.

2. *Труш В.* Доклад на конференции молодых ученых по компьютеризации образования. Нарва, апрель 1987 г.

3. *Гельтищева Е., Селихова Г.* Режим работы за дисплеем // ИНФО. 1987. № 1. С. 82—84.

4. *Антипов И., Колобов С., Ермолаев О., Анисимов В.* О методике психологического исследования адаптации школьников к учебной нагрузке при работе на компьютере // ИНФО. 1988. № 2. С. 71—73.

5. *Гиглавый А., Гуткин М.* Тормоза компьютеризации // Учит. газета. 1988. 10 марта.

6. *Садовская Н. А.* Компьютеры и среднее образование: проблемы, проблемы, проблемы... // Автоматизированные системы научных исследований, обучения и управления в вузах: Межвузовский сборник научных трудов. Новосибирск, 1986. С. 51—63.

7. *Галкина В.* Эта упрямая информатика // Учит. газета. 1988. 28 апр.

8. *Каптелинин В. Н.* Психологические проблемы формирования компьютерной грамотности школьников // Вопросы психологии. 1986. № 5. С. 54—65.

9. *Смолян Г. Л.* Электронно-вычислительная машина и человек // Коммунист. 1985. № 1. С. 105—106.

О. КРОНЧЕВА

От редакции. Мы предложили прокомментировать статью О. Крончевой кандидату медицинских наук Е. Гельтищевой, занимающейся этими проблемами.

Введение в общеобразовательные школы предмета «Основы информатики и вычислительной техники» действительно не было согласовано с Минздравом СССР. Сразу после введения этого предмета в средние общеобразовательные школы врачи-гигиенисты включились в исследование (с использованием физиологических, психологических и других методов) условий, организации рабочего места и режима работы за видеотерминалами. Врачами различных специальностей были проведены целенаправленные исследования по изучению состояния здоровья учащихся IX класса, у которых по плану учебных занятий на информатику отводился один урок в неделю. Данные исследования [1] позволили прийти к выводу об отсутствии отрицательного действия ЭВМ на состояние здоровья. К аналогичным выводам пришли и мы при изучении функционального состояния и состояния здоровья наблюдаемых учащихся IX—X классов. Если в отдельных случаях и имелось ухудшение здоровья, то объяснить его только работой учащихся на видеотерминалах в течение 1—2 уроков в неделю нельзя, поскольку общая учебная нагрузка в IX—X классах очень большая. В то же время проведенные физиологические и психологические исследования учащихся во время уроков информатики позволяют выявить отдельные отклонения со стороны высшей нервной деятельности, центральной нервной системы, зрительного анализатора и сердечно-сосудистой системы, и степень их определяется воздействием комплекса факторов при работе на видеотерминалах без соблюдения гигиенических требований и профилактических мероприятий.

В радиационном отношении выпускаемые видеотерминалы не представляют опасности для здоровья. Неиспользованное рентгеновское излучение значительно ниже допустимых санитарных норм (100·мкР/ч) [2], [3].

В настоящее время для учащихся разработаны режимы работы в зависимости от класса используемых видеотерминалов [4], определены условия работы и организация рабочего места [5], [6] с учетом современных требований.

Что касается студентов, то впервые исследования по изучению влияния учебных занятий с применением дисплеев были начаты в МНИИ гигиены им. Ф. Ф. Эрисмана в 1982 г. На основании многолетних исследований в 1988 г. Минздравом СССР и Минвузом СССР утвержден документ, регламентирующий условия и работу студентов за видеотерминалами, организацию рабочего места и режим занятий. В нем также отражены современные требования к видеотерминальным устройствам с учетом международных данных и меры профилактики, направленные на охрану здоровья. Соблюдение этих требований при работе с видеотерминалами позволит сохранить хорошую работоспособность и предупредить развитие переутомления, которое, накапливаясь в организме, может привести к срыву адаптационных процессов и развитию заболеваний.

Исследования по изучению работы учащихся и студентов на видеотерминалах различных типов продолжаются в институтах медицинского, педагогического и технического профиля, уточняются отдельные положения.

В настоящее время подготавливаются санитарные правила при работе с дисплеями, которые будут едины для учащихся средних общеобразовательных школ, ПТУ, студентов техникумов и вузов, а также для взрослых, работающих с видеотерминалами. В них должна найти отражение и работа педагогов общеобразовательных школ и вузов.

Литература

1. *Ананьева Н. А., Неверова Л. Г.* Работа на дисплеях и состояние здоровья // Гигиена и санитария. 1987. № 6.

2. Нормы радиационной безопасности. НРБ-76. М.: Атомиздат, 1978.

3. Нормы радиационной безопасности. НРБ-76 и Основные санитарные правила ОСП — 72/80. М.: Энергоиздат, 1981.

4. Режим занятий учащихся за видеотерминалами. (Временные рекомендации). Утверждены Минпросом СССР 05.02.87, № 295—38/14, и согласованные с Минздравом СССР 27.01.87.

5. Кабинет вычислительной техники всех типов средних учебных заведений (на базе персональных микро-ЭВМ): Методические рекомендации. М., 1986. 43 с.; Информатика и образование. 1986. № 3.

6. Методические указания по профилактике переутомления студентов вузов при работе с видеотерминалами. Утв. Минздравом СССР 05.03.88, № 4563—88, и Минвузом СССР 09.03.88, № 96—21—05/65.

Е. ГЕЛЬТИЩЕВА,
ст. научный сотрудник МНИИ гигиены им. Ф. Ф. Эрисмана

Конкурс «на перекрестке»

Населенный пункт был небольшим. Пять прямых, как стрела, улиц пересекали четыре, образуя 12 кварталов. На некоторых перекрестках висели дорожные знаки. На площади, имеющей номер 2, поблескивая никелем и свежевыкрашенными боками, собрались тяжелогруженые машины. Каждая вместе с грузом весила 8 тонн. 18 автомобилей замерли у стартовой черты. Судья поднял руку...

В начале октября прошлого года необычное оживление царило в Институте усовершенствования учителей в болгарском городе Варна. Стайки ребят тянулись к классу, в котором были установлены персональные компьютеры. То у одного, то у другого монитора собирались разноязычные группы и обсуждали на только им понятном языке возникающие при работе на ЭВМ ситуации. Огромные щиты при входе в институт объясняли происходящее — начинался международный конкурс учащихся профтехучилищ социалистических стран «Микрокомпьютер: программирование и эксплуатация». Ребята из Кубы, ГДР, Польши, Венгрии, Советского Союза и Болгарии собрались, чтобы помериться силами в умении составлять и отлаживать программы, обосновывать выбранный метод решения.

...Взревели моторы. Взмах флажка. Гонка началась. Цель — перекресток номер 15. «Кирпич», ограничение веса автомобиля пятью тоннами, поворот направо, запрещение поворота налево — такие дорожные знаки должны были направлять каждую машину по счастливому для нее пути к финишу. К сожалению, не всем удалось уйти со старта...

Торжественное открытие конкурса. Участники получают дискеты и все необходимое для соревнования. Еще раз обсуждается порядок работы ребят и международного жюри, в которое вошли все руководители команд.

Наступил день состязания. Последние слова напутствия ребятам говорят руководители команд: до окончания соревнования они больше не увидятся со своими «подопечными». У жюри началась кропотливая работа по выбору конкурсной задачи и определению критериев ее оценки. Какой из предложенных командами задач отдать предпочтение? Наиболее соответствовали условиям конкурса задачи болгарской и советской делегаций. После жарких споров мнение членов жюри склонилось в пользу болгарской. Оперативный перевод ее на рабочие языки. Оговорены принципы оценки решений. Все готово к состязанию.

...У перекрестка грузовики останавливаются в нерешительности: какую дорогу выбрать? Поехать направо? Но туда проезд запрещен. Четыре поворота налево — и снова окажешься на прежнем месте. Непросто выбрать один из возможных путей. Разъехавшиеся по всему городку машины иногда нарушали правила движения, но бесстрастная техника фиксировала все нарушения...

В трех аудиториях за рабочими столами с компьютерами IBM PC и «Правец-8М» притихли ребята. Условия задачи и правила оценки решения переводчики сообщают каждому участнику. Четыре часа, отведенных на работу, промелькнули как одно мгновение. На следующий день руководители команд «сдавали» программы своих ребят координационной комиссии, в которую вошли сотрудники научных организаций Болгарии. Эта комиссия, а не международное жюри приняла решение о победителях.

...Один из грузовиков, словно на ощупь, анализируя пройденный путь, неуклонно приближался к цели. Осталось объехать последний квартал, вот он финиш. И только машина с красно-белым флажком на капоте, постоянно возвращаясь к стартовому перекрестку, уверенно доезжала до финиша, двигаясь при этом не только по правилам, но и каждый раз новым маршрутом...

И вот началось официальное закрытие конкурса. Объявляются победители: учащийся техникума из Болгарии Георги Ривов и поляк Марцел Войяс. Точное и правильное выполнение всех условий конкурса, структурный подход к программированию, рациональное использование структур данных — вот что выгодно отличало их работы. Награда соответствовала теме конкурса: бытовые компьютеры «Правец-8Д». Отдельные неточности в реализации правильного алгоритма не позволили учащемуся киевского училища Руслану Кондрацкому подняться выше третьего места.

Приятной неожиданностью для собравшихся было сообщение руководителя советской команды о награждении Почетным призом Госкомитета СССР по народному образованию «За оригинальную программу» будущего рабочего из Кракова М. Войяса. Четыре матрешки с необычными именами — Basic, Pascal, Logo и Assembler привезли из СССР фотоаппарат «Зенит», который поможет ему не забывать, что в жизни существуют не только компьютерные «миры».

...Устало замерли на дорогах городка автомобили. Будет ли следующий конкурс? Будет ли изматывающая гонка друзей, которые пусть не надолго, но стали конкурентами?

Наступил день отъезда. Жаркое болгарское солнце влекло к спокойному, прохладному морю. Но Эдик Зотов, который учится в 148-м училище Москвы, предпочел остаться в компьютерном классе, чтобы обменяться новинками системного программного обеспечения с болгарскими ребятами, а киевлянин Игорь Давидович беседовал с кубинцами о том, как живет, учится и отдыхает молодежь. Уже отправлены первые письма Диляне — десятикласснице из специализированной варненской школы с углубленным изучением иностранных языков, которая много сделала, чтобы познакомить советскую команду с достопримечательностями уютного портового города, чтобы советские ребята чувствовали себя свободно в гостеприимной Болгарии. У всех участников появились новые друзья. И в этом одно из главных достижений конкурса.

Представители всех стран были единодушны в том, что конкурс может стать традиционным, но формы его проведения должны совершенствоваться. Может, следует предлагать ученикам не одну, а несколько задач разной сложности? А если ребят из разных стран включить в бригады и предложить им разработку программ общесистемного обеспечения или учебного назначения, имеющих практическую ценность? А как относиться к предложению провести фестиваль по информатике, во время которого будут и конкурс, и семинары, и лекции известных специалистов, знакомство с учебными заведениями, культурная программа? Только практика многосторонних контактов поможет выбрать наиболее действенные формы сотрудничества.

В. СОКОЛОВ

Семинар в Симферополе

В сентябре 1988 г. в Симферополе проходил IV Всесоюзный семинар «Разработка и применение программных средств ПЭВМ в учебном процессе». Организаторы семинара — ИПИ АН СССР, Госкомитет СССР по народному образованию, ИК им. Глушкова АН УССР, СГУ. На семинаре присутствовали более 120 специалистов. К началу семинара был выпущен сборник докладов (ИПИ АН СССР, тираж 400 экз.).

Результаты исследований и разработок в области учебной информатики, выполненных со времени проведения III Всесоюзного семинара (г. Пущино), были доложены на заседаниях пяти секций: «Теоретические аспекты создания систем учебной информатики», «Инструментальные средства систем учебной информатики», «Программная поддержка курса ОИВТ», «Новые применения ВТ в учебном процессе и их программная поддержка», «Интеллектуальные системы в обучении». За «круглым столом» участники семинара провели дискуссию: «Нужен ли нам Пролог?»

На семинаре было отмечено, что не все рекомендации III Всесоюзного семинара были выполнены. До сих пор не решен вопрос об авторских правах на программные сред-

ства, не представлены предложения по архитектуре и базовому программному обеспечению учебных ПЭВМ. Участники семинара обратили внимание на необходимость усиления фундаментальных исследований в области создания перспективных средств ВТ, программного обеспечения и массового внедрения результатов исследования в систему образования всех уровней, на необходимость координации работ в области учебной информатики.

Рекомендации IV Всесоюзного семинара «Разработка и применение программных средств ПЭВМ в учебном процессе»

1. Просить Госкомитет СССР по народному образованию, ГКВТИ СССР рассмотреть вопрос о первоочередных поставках инструментальных комплектов учебной вычислительной техники (КУВТ) MSX-2, УКНЦ и «Корвет» разработчикам программных средств и научным организациям, ведущим исследования в области учебной информатики.

2. Обратить внимание ГКВТИ СССР, Минэлектронпрома СССР, Минрадиопрома СССР на серьезное отставание в оснащении учебных заведений средствами вычислительной техники, ее низкое качество, в частности на недопустимость поставки КУВТ-86 на базе БК-0010 в учебные заведения, поскольку по своим технико-программным возможностям он не отвечает требованиям, предъявляемым к учебным КУВТ.

3. Выдвинуть в качестве первоочередной задачи разработку новых перспективных моделей ПЭВМ, так как развитие перспективных систем учебной информатики сдерживается ограниченными возможностями выпускаемых в настоящее время учебных ПЭВМ.

4. Обратить внимание Госкомитета СССР по народному образованию на следующие первоочередные работы для финансирования:

разработку разнообразных обучающих средств для пользователей-непрофессионалов в области ВТ;

стандартизацию и унификацию систем учебной информатики;

разработку и внедрение «электронной доски»;

разработку коммуникационных средств с привлечением соответствующих министерств и ведомств;

разработку перспективных информационных технологий для сферы образования, в частности видеокомпьютерных систем, компьютерных лабораторий и т. п.

5. Необходимо предусмотреть разработку средств для обеспечения учебного процесса в национальных школах (возможность работы на языках народов СССР), а также разработку аппаратно-программных средств, не предусмотренных в серийно выпускаемых ПЭВМ, для профессионального обучения в средних специальных учебных заведениях, ПТУ, системе повышения квалификации.

6. Учитывая сложившуюся в стране обстановку, разобщенность исследований, проводимых различными организациями в области разработки и применения систем учебной информатики, необходимо создать межведомственный координационный центр, в задачи которого входило бы:

обобщение опыта создания и применения средств учебного назначения;

критический анализ разрабатываемых и внедряемых средств учебной информатики;

установление приоритетности разработки систем учебной информатики;

выявление перспективных направлений использования новых информационных технологий в учебном процессе;

координация работ, проводимых различными ведомствами;

определение единой технической политики в области использования новых информационных технологий в образовании.

7. Рекомендовать Госкомитету СССР по народному образованию усилить исследования в области психолого-педагогических проблем, разработки и применения новой информационной технологии в обучении. В связи с этим целесообразно провести отдельный всесоюзный семинар по психолого-педагогическим аспектам систем учебной информатики.

8. Считать целесообразным и полезным периодическое проведение настоящего семинара. Провести V Всесоюзный семинар «Разработка и применение программных средств ПЭВМ в учебном процессе» на базе Северо-Осетинского госуниверситета г. Орджоникидзе.

Примечание. Оргкомитет семинара, при условии достаточного количества предварительных заказов, готов организовать дополнительный тираж и выслать желающим сборник докладов по договорной цене. Заказы направлять по адресу: 117900 ГСП-1, Москва, ул. Вавилова, 30/6, ИПИ АН СССР, отдел учебной информатики.

Конференция в Казани

Если верить законам диалектики, мы стоим на пороге качественного скачка в области компьютеризации образования. По крайней мере количество уже налицо: осень 1988 г. стала сезоном конференций, посвященных этой проблеме. Даже при всем желании организаторы конференций едва ли сумели бы так скоординировать сроки, чтобы перед участниками не возникала проблема выбора.

Если по своему формальному статусу — республиканская — конференция в Казани в октябре 1988 г. должна была бы занять весьма скромное место среди прочих, то по составу участников ее можно назвать всесоюзной и отнести к числу наиболее представительных конференций года. И это несмотря на то, что многие ведущие специалисты в области педагогики и психологии из Москвы и Киева приехать не смогли.

Конференция была организована АПН СССР совместно с НИИ профтехпедагогики АПН СССР, Татарским ИУУ и Казанским университетом. Хочется отметить активное участие в подготовке и проведении конференции Д. М. Шакировой, Л. С. Волковой.

Отличительной особенностью конференции было активное участие в ее работе учителей информатики школ и ПТУ Татарии наравне с иногородними участниками и доцентами казанских вузов. Учителя были единодушны в своей оценке — конференция принесла им огромную пользу.

Тем не менее хочется отметить следующее.

По оценкам специалистов на конференции было заметно больше слов, чем демонстраций программ, и практически не было их обмена. Большой наплыв участников — несомненный признак интереса к заседаниям, однако именно из-за многолюдности программы демонстрировались как кино, а возможность обсуждения их «начинки» практически исключалась. Можно высказать и упрек в адрес организаторов: в первые дни машин для демонстрации программ вообще не было, а в последующие их явно не хватало (ни по количеству, ни по ассортименту).

Кроме того организаторы не справлялись с простой арифметической задачкой: на двухчасовое пленарное заседание было запланировано семь (а с учетом неявившихся участников — девять) докладов по 30—40 мин. И если на интересные, но носящие обзорно-исторический характер доклады руководителей Министерства народного образования ТатАССР, НИИ профтехпедагогики и Центра ИВТ при Казанском университете времени хватило, то последующие проблемные доклады были скомканы до кратких сообщений.

Но именно педагогические проблемы оказались в центре внимания на пленарном заседании. Два доклада были специально посвящены воспитательной работе на уроках информатики в школах и ПТУ.

«Что вы знаете об ASP — corporation?» — с этого интригующего вопроса начал А. Н. Гриценко из Барнаула свой рассказ о хайкерах (от английского «high»), которых не следует путать, но очень трудно не перепутать с хаккерами. Хайкерами называют себя ребята Андрей, Сережа, Паша (ASP), которые успешно разрабатывают программы для БК-0010. Большой интерес участников методической секции вызвали сообщения В. В. Дубининой из Казани об эстетическом воспитании на уроках информатики и И. П. Фроловой из Нижнего Тагила об использовании ПЛС-3. А на другой секции наибольшее внимание привлек доклад В. Б. Доронина из Киева об Р-технологии программирования.

Секционные заседания были необычно многолюдными (около 70 участников на каждой секции), и это привело к бурным дискуссиям. Извечными становятся на конференциях дебаты вокруг больного вопроса о распространении и тиражировании программных продуктов. Что важнее: выделить школам финансы и дать право самостоятельного приобретения программ либо централизованно собирать и распространять их? В конце концов дискутирующие стороны соглашаются, что нужно и то и другое. Пока же ни того, ни другого нет. «Но ведь есть же фонды алгоритмов и программ»,— спохватывается кто-то. «Да, но нам просто предлагают присылать программы, а авторство не сохраняют,— возражает Д. М. Шакирова,— и потом мне самой же предлагали купить мою собственную программу за тысячу рублей». Сходные примеры вспоминают и другие участники. Действительно, не слишком ли разорительно для органов образования бесплатно отдавать программы НИИ ИВТ и различным фондам, а затем покупать у них же за большие деньги?

Не менее острым было и обсуждение «проблемы лаборантов». Те, у кого их пока нет, сетовали на возникающие трудности. Те же, у кого лаборанты есть, говорили,

что из-за низкой квалификации они только мешают. После обсуждения участники конференции включили в резолюцию следующие предложения.

1. Предоставить право учебным заведениям, включая школы, самостоятельно решать вопрос о приобретении необходимых им программных продуктов. Обеспечить финансирование соответствующих расходов.

2. Обеспечивать дисплейные классы дискетами из расчета не менее 3М байта на каждое рабочее место ученика, не считая централизованно поставляемой библиотеки стандартных программ.

3. Организовать тиражирование лучших программ, созданных преподавателями школ и вузов, в том числе в МГУ (руководитель А. Г. Кушниренко), МИЭМе (В. А. Каймин), Казанском университете (А. Н. Таркаев), Шуйском пединституте (А. Г. Васильченко), Ярославском пединституте (В. М. Ермакова), Алтайском краевом ИУУ (А. Н. Гриценко). Приобрести для последующего тиражирования лучшие программы, созданные в учреждениях, не подчиненных ГКНО СССР, в том числе пакеты «Роботландия» (ИПС АН СССР, г. Переславль-Залесский) и «Надежда» (НТТМ «Ритм», г. Новосибирск).

4. Преподавателям вузов планировать разработку педагогических программных продуктов как часть их работы. Приравнивать зарегистрированную и внедренную учебную программу к публикации научной или методической статьи.

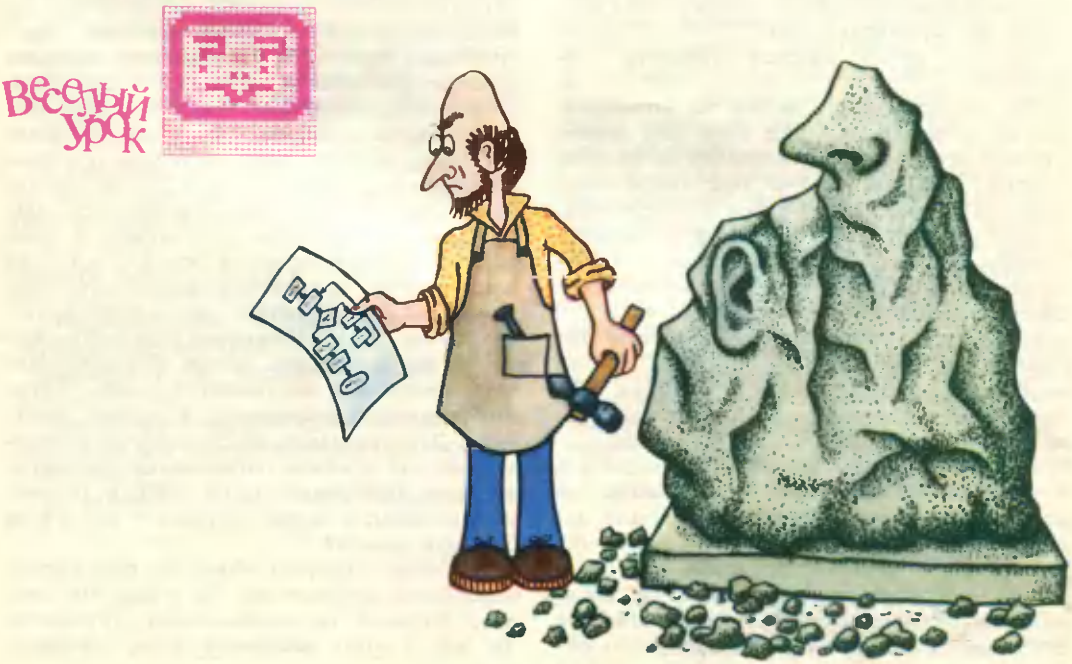
5. Создать в системе ГКНО СССР сеть центров по сбору информации об имеющихся учебных программах и их тиражированию. Головные и региональные центры этой сети оснастить модемами, позволяющими передавать программы с одних типов ЭВМ на другие. Наряду с ними должны быть специализированные центры, ориентированные на конкретный тип ЭВМ или язык программирования.

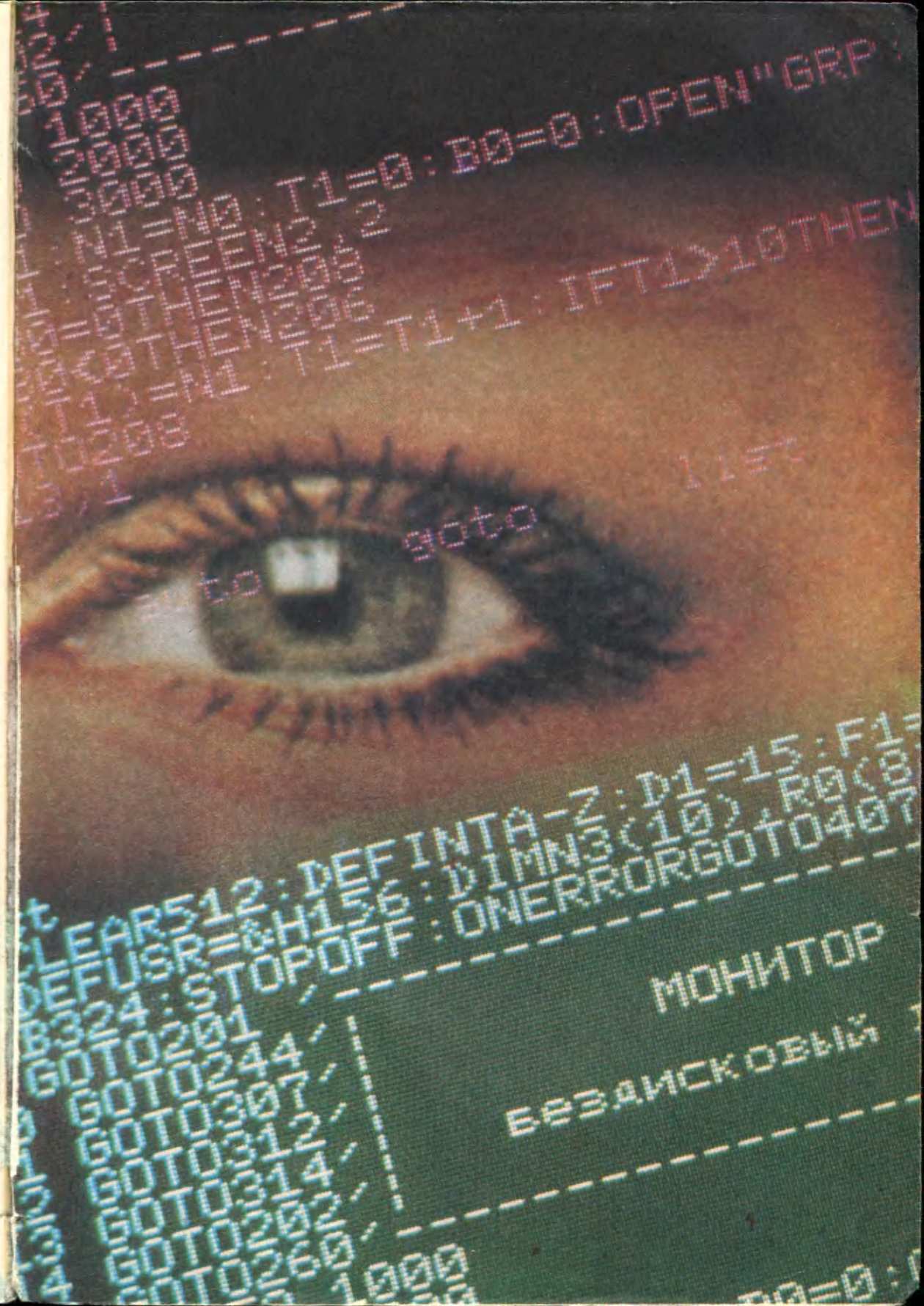
В качестве основных центров сети рекомендуются Казанский, Ленинградский, Московский и Новосибирский университеты; Свердловский, Харьковский и Ярославский пединституты; Алтайский краевой, Московский областной и Татарский республиканский ИУУ; Дом компьютерной техники Ивановского гороно; Ленинградский Дворец пионеров; Вильнюсский Дом юного программиста; средняя школа № 82 АПН СССР (пос. Черноголовка Московской обл.); интернациональная школа-интернат им. Е. Д. Стасовой (г. Иваново); школа № 6 (г. Переславль-Залесский).

6. Включить в штат ИУУ и школ, имеющих дисплейные классы, ставку лаборанта кабинета вычислительной техники.

Участники конференции ждут от Госкомитета СССР по народному образованию конкретных действий по реализации этих предложений.

В. ФЕДОТОВ




МОНИТОР
БЕЗДИСКОВЫЙ

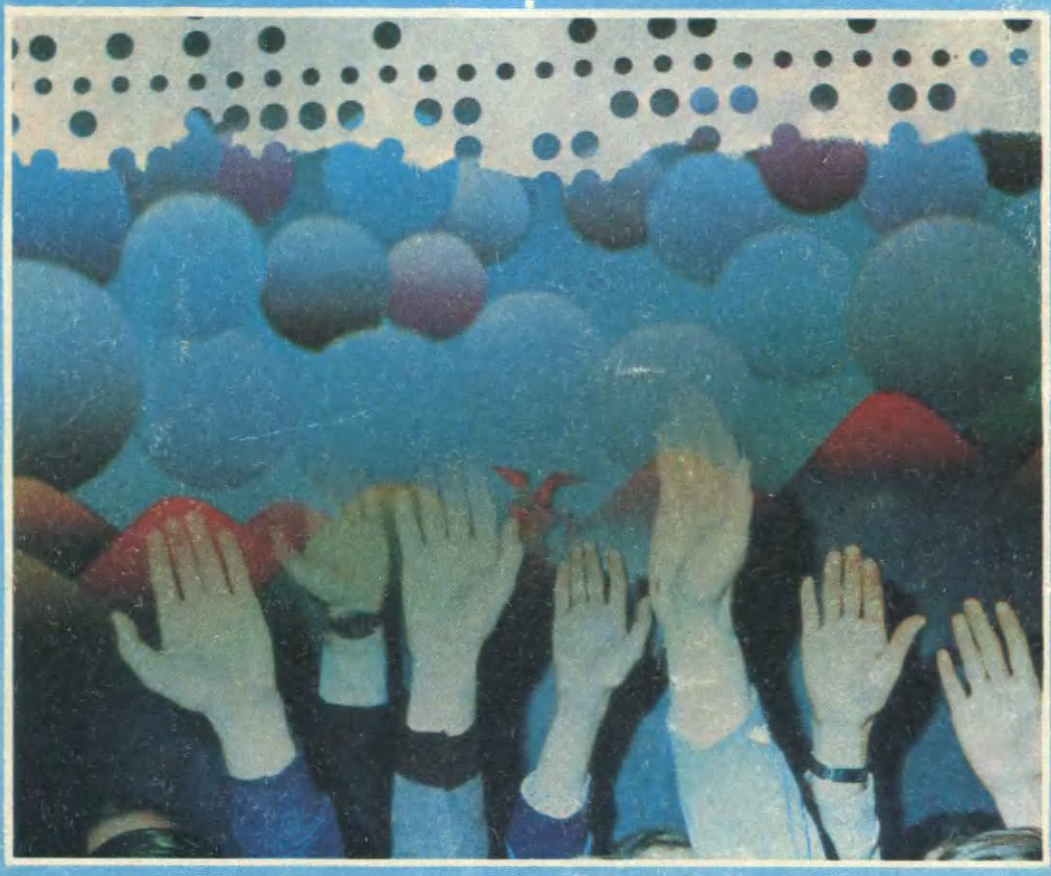
Б Кит 11-41

Цена 60 коп.
70423

ИНФО

1'89

ИНФОРМАТИКА
И ОБРАЗОВАНИЕ



Информатика и образование, 1989, № 1, 1—128