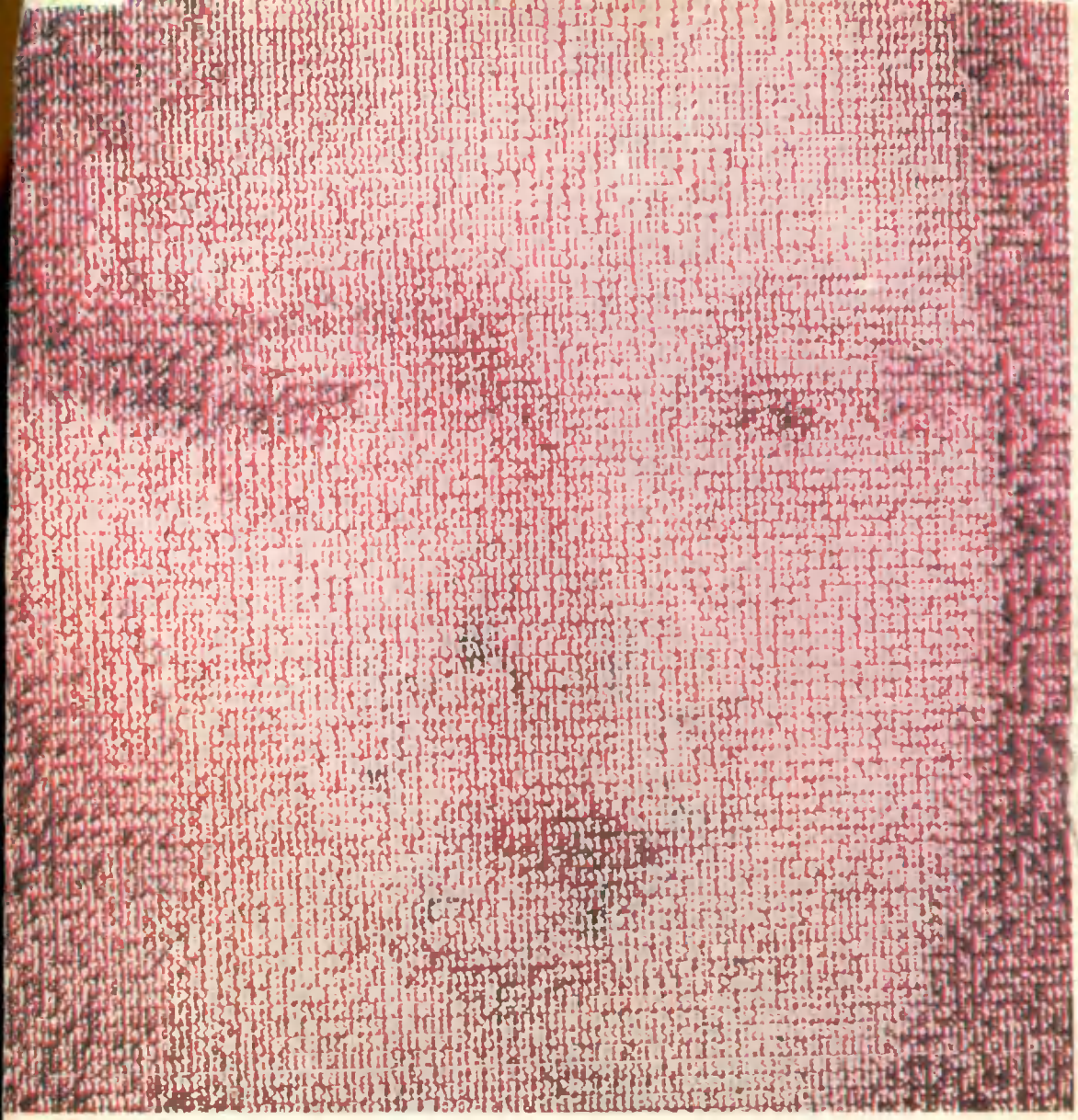
ISSN 0234—0453

ИНФО

ИНФОРМАТИКА И ОБРАЗОВАНИЕ

1 1987



Рисунки на экране

С помощью ЭВМ получают удивительные изображения, которые другими способами выполнить невозможно.

Машинная графика играет немалую роль в изучении машинного варианта основ информатики и вычислительной техники. Специальные языки программирования для начинающих, например Лого, основаны на построении изображений на экране дисплея. Это позволяет школьникам быстро усвоить понятия алгоритма, составлять простейшие программы.

С помощью языков высокого уровня компьютер может рисовать сложные изображения (см. фото на обложке), используя цветовую гамму.

Научно-методический журнал
Министерства просвещения СССР,
Государственного комитета СССР
по профессионально-техническому образованию,
Министерства высшего
и среднего специального образования СССР

Выходит шесть раз в год
Издается с 1986 г.

Издательство «Педагогика»
Москва

ИНФОРМАТИКА И ОБРАЗОВАНИЕ

1'87
январь — февраль

Содержание

Репортаж номера

Педагогический опыт

Нам пишут

Информация

Веселый урок

Редактор отдела *К. Шеховцев*
Научный редактор *Т. Драгныш*
Заведующая редакцией *Н. Игнатова*
Художественный редактор *Л. Розанова*
Корректор *Н. Минервина*

Цветные фото на обложке — Э. Бажилина.
В оформлении номера принимали участие Р. Бумагин, С. Весков, Ю. Весков, С. Кустарева, И. Печенев, А. Пономарев, В. Сазонов.

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

Почтовый адрес: 107847 Москва, Лефортовский пер., 8

Сдано в набор 01.12.86. Подписано в печать 31.12.86. А 08781. Формат70×100/16. Печать офсетная. Усл. печ. л. 10,40. Уч.-изд. л. 12,19. Усл. кр.-отт. 42,88 Тираж 87 180 экз. Заказ 3409. 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.

В. ТЕРСКИХ

Автоматизированные системы: концепция промышленного изделия

Предлагаемая вашему вниманию статья о вещах, к сожалению, не ставших пока для нас обыденными. Системный подход к социальным, экономическим, производственным проблемам только начинает прокладывать себе дорогу. Термины «изделие», «обеспечение», «система» и другие в данном контексте используются не в привычных своих значениях.

Изделие здесь — результат некоей деятельности, существующий в форме, доступной для использования, имеющий социально-экономическую ценность; изделием может быть, например, приказ о введении более современной организации трудового процесса.

Эффективность изделия определяется отношением всех затрат (на изготовление, передачу, поддержание работоспособности и т. д.) ко всем выгодам — социальным, экономическим, психологическим и т. п.

Жизненный цикл изделия — период от момента зарождения идеи изделия до исчезновения потребности в нем. Включает стадии разработки, изготовления, передачи, эксплуатации изделия.

Надеемся, что высказываемые автором мысли заинтересуют наших читателей.

Разработчики средств автоматизации должны создавать программы для ЭВМ как продукцию производственно-технического назначения (для краткости далее будем использовать термин «изделие»). Это решение отражает серьезные изменения в подходе к автоматизации народного хозяйства, темпы развития которой требуют прекращения анархии в выпуске программ, поскольку при этом расходуется все больше средств на дублирование разработок без твердой гарантии результата. Быть может, с дублированием можно справиться, устранив организационные трудности в передаче результатов разработки? Скорее всего, оно имеет под собой более прочную основу. Действительно, в условиях фактически бесконтрольной адаптации зарубежного и копирования отечественного программного обеспечения тем не менее постоянно инициируются оригинальные разработки. Причина, на наш взгляд, состоит в том, что налицо существенная разница между специализированной программой, доступной для конечного пользователя-непрограммиста, и более гибкой системой; для эксплуатации последней необходимо обладать программистской квалификацией. Иными словами, для реакции на изменения в потребностях конечного пользователя или даже для осуществления (в большой программе) нестандартной операции нужен профессиональный программист. Кроме того, достаточно сложная программа, увы, не может быть отчуждена от автора — ввиду неразвитости технологии создания программного обеспечения сопровождать программу

(вносить грамотные изменения) может лишь он.

Что мы видим в окружающей нас действительности? Программист создает программу для конкретного пользователя и при этом ориентируется на ее включение в имеющуюся под рукой программно-аппаратную среду, которую без большой натяжки можно считать неповторимо-индивидуальной. Документирована программа кое-как, ведь для этого не хватает времени, желания и специальных знаний. Модификация с целью устранения ошибок и расширения возможностей программы производится методом «наложения заплаток», а поэтому через некоторое время сам автор уже не может разобраться в ней. Если программа передана еще кому-то, то о произведенных модификациях ему, как правило, ничего не сообщают. А если бы и сообщили? У автора к этому времени много другой работы, а службы сопровождения и документирования отсутствуют.

Институты, проектирующие программные средства и аппаратуру, заводы, выпускающие устройства и ЭВМ, монтажно-наладочные и обслуживающие организации выполняют работы не намного лучше. Сопровождение, поддержка и некоторые виды испытаний практически отсутствуют. Программы и аппаратура модифицируются независимо друг от друга; пользователи об этих изменениях узнают только случайным образом.

Существующее положение неприемлемо — программа привязана и к автору, и к организации. Нужна система, с помощью которой грамотный пользователь, не будучи программистом, смог бы создать продукт для автоматизации своей деятельности.

Для того чтобы программу можно было распространять, необходимо при разработке учесть требования потенциальных пользователей, влияние программного и аппаратного окружения, в котором ей придется работать, а также возможные конфликтные ситуации в данных, программах и аппаратуре. Необходимо предусмотреть средства диагностики программы, восстановления при сбоях, испытания на соответствие исходным требованиям, разработки и внесения изменений в функционирующую программу.

Программа должна быть документирована с использованием стандартизированных форм документов и терминологии. Пользователям и обслуживающему персоналу необходимо предоставлять информацию о свойствах программы и возможностях ее приобретения, оказывать помощь в освоении и эксплуатации.

Не всегда осознается, но всегда фактически присутствует постулат, что программист энциклопедически эрудирован и может решить все вопросы, возникающие в процессе автоматизации. На самом деле программисты не в состоянии даже поставить большинства вопросов, так как не обладают достаточным объемом знаний. Никто не обучает программиста психофизиологическим основам построения интерфейса между человеком и системой, правовым основам использования информации, организационным и экономическим механизмам управления и другим специальным дисциплинам. Естественно, что ничего кроме кустарных результатов, основанных на интуиции программиста, ожидать не приходится.

Ситуация с созданием автоматизированных систем еще хуже. Она напоминает времена кустарного радиолюбительства, когда желающий послушать радиопередачу должен был изучить радиоделло, достать дефицитные детали, разработать и собрать радиоприемник, затем бесконечно ремонтировать его и т. д.

В ГОСТах и ОРММ (общеотраслевых руководящих методических материалах) определены виды обеспечения автоматизированных систем (рис. 1): информационное, эргономическое, лингвинистическое, программное, техническое, организационное, методическое и математическое. Следовало бы добавить еще один вид обеспечения — кадровое, а организационное расширить до организационно-правового. Кадровое обеспечение определим как совокупность всех специалистов, профессиональная деятельность которых имеет отношение к рассматриваемой автоматизированной системе.

5

1

В настоящее время делаются попытки довести программные и технические (аппаратные) компоненты до уровня изделий. Более того, специалисты приходят к пониманию того, что минимальным компонентом системы должны быть программно-аппаратные изделия, рассматриваемые как единое целое. О том, чтобы распространить концепции изделия (продукции производственно-технического назначения) на остальные виды обеспечения, всерьез еще не думали.

Меньше всего повезло кадровому обеспечению. В создании системы и обеспечении ее жизненного цикла кроме администраторов, программистов и электронщиков должны принимать участие юристы, экономисты художники, психологи, социологи. Но попробуйте принять на работу юриста-разработчика, экономиста-разработчика! Кто пытался это сделать, знает, что найти таких специалистов сложно, а принять на работу без нарушения трудового законодательства и с сохранением профессионального статуса невозможно.

Сегодня формирование кадрового обеспечения представляет собой смесь кустарщины и нарушений штатного расписания.

Очень плохо обстоит дело с обучением. Учить специалистов можно только на примере существующих и в процессе создания новых систем-изделий, а они не появятся, пока не будет соответствующих специалистов.

Некоторые считают, что в других странах ситуация лучше. Отсюда массовые закупки зарубежных персональных компьютеров, адаптация программных средств, попытки воспроизвести фрагменты экспериментов по обеспечению жизненного цикла систем и т. п. Это ложный и опасный для отечественной автоматизации путь. Принимать положительный опыт, конечно, необходимо, но! Во-первых, фрагменты *чужой* технологии приносят лишь разочарование именно потому, что из осколков двух ваз нельзя собрать салатницу. Во-вторых, зачастую чужие средства и технология требуют воспроизведения капиталистических отношений на советской почве. И наконец, за рубежом не все так бла-

гополучно, как кажется со стороны, особенно в автоматизации интеллектуальной деятельности. Надежность и качество средств автоматизации выше, насыщенность ими рынка больше, обязательства перед покупателем соблюдаются лучше, время на разработку и реализацию меньше, а следовательно, и реакция быстрее; тем не менее в профессиональной литературе постоянно задаются тревожные вопросы: почему «белые воротнички» отказываются работать с системой? Почему профессиональные персональные компьютеры не дали ожидаемого эффекта? И т. д.

Одни видят причину в несовершенстве обеспечения жизненного цикла систем. Отсюда многочисленные эксперименты с «бригадами главных программистов», проектированием «сверху вниз», организацией сопровождения и поддержки. Другие высокомерно возлагают ответственность на конечных пользователей и призывают их преодолеть страх перед компьютером, овладеть второй грамотностью, т. е. программированием, и т. п.

Устойчивых положительных результатов пока нет.

Причина, на наш взгляд, в том, что проблема решается недостаточно комплексно. В основном занимаются созданием автоматизированных систем «при организациях», а не созданием автоматизированных организаций, органически включающих в себя системы. Недостаточно четко определено само понятие систем-изделий. Отсутствует классификация программных, программно-аппаратных изделий и других видов обеспечения, учитывающих потребности пользователей и обеспечивающих эффективность изделий в целом на протяжении всего жизненного цикла. Отсутствует классификация специалистов, описание их должностных обязанностей и прав, обоснованное нормирование трудозатрат. Отсутствует технология индустриального обеспечения жизненного цикла систем с активным участием в нем конечных пользователей. Отсутствует комплексная организационно-правовая база для интеграции усилий всех организаций и специалистов, участвующих в обеспечении жизненного цикла. Не создана инфраструктура обеспечения изделий.

Эффективная автоматизация может осуществляться только при комплексном решении вышеперечисленных проблем и ряда других, которые возникнут в процессе.

Попытаемся проанализировать аспекты создания изделий, которые следует рассматривать как совокупность взаимосвязанных проблем и решать в комплексе, если мы хотим, чтобы автоматизация стала действенным инструментом ускорения социально-экономического развития страны и научно-технического прогресса.

К этим аспектам относятся:

организация принятия решений;

структура изделия;

функции службы жизнеобеспечения изделия;

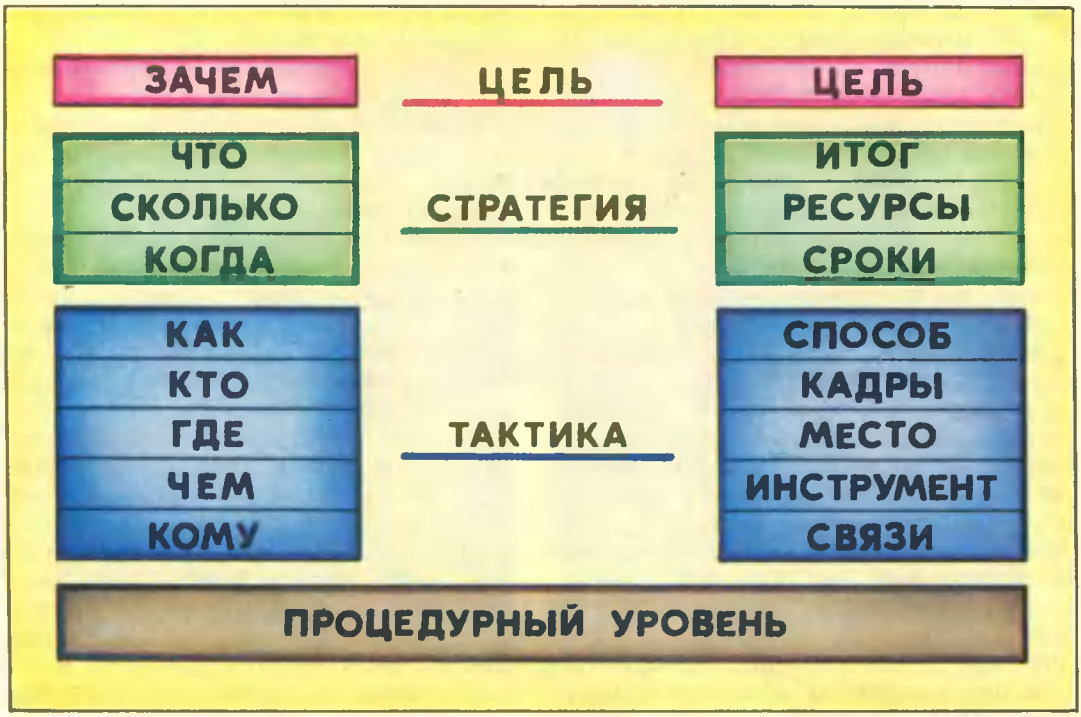
жизненный цикл изделия;

технологические уровни автоматизации деятельности;

организация междуведомственной кооперации в обеспечении изделий.

Организация принятия решений

Создание любого изделия начинается с принятия соответствующего решения, которые должны приниматься и впоследствии — на всем протяжении жизненного цикла, на самых разных уровнях, касающиеся всех компонентов и изделия в целом. Естественным требованием для оптимизации принятия и исполнения решений является наличие необходимой информации (соответствующих документов) и отсутствие ненужной. Многочисленные попытки создания простых и эффективных средств обеспечения деятельности необходимым и достаточным набором документов не дали результатов. Предлагаемые средства еще не используются регулярно и повсеместно. Причинами этого (в числе прочих) может быть то, что требования ГОСТ, рекомендации ОРММ и предложения исследователей не сведены в комплексную организационно-правовую систему, не обеспечены обучение и контроль за их использованием, а также то, что не материализованы в качестве изделий соответствующие виды обеспе-



2

чения (см. раздел «Структура изделия»).

Для анализа воспользуемся упрощенной мнемосхемой набора вопросов, на которые необходимо ответить в процессе принятия решений (рис. 2). Ей же воспользуемся для систематизации наших рассуждений.

Исследователи в области организации деятельности, как правило, выделяют в управлении четыре уровня принятия решений: цель, стратегия, тактика и процедурный уровень.

На верхнем уровне определяются цели деятельности. Решения этого уровня являются ответами на вопрос «Зачем?»; например: «Зачем создается изделие?» Или при проектировании системы: «Зачем система формирует такой-то продукт и обеспечивает его жизненный цикл?»

Видов целей четыре: научно-технические, производственные, экономические и социально-политические.

Ответы формулируются в терминах «повышение эффективности», «ускорение темпов роста» и т. п. В такой ситуации критерии эффективности естественно и логично вытекают из принятого определения цели.

На практике, к сожалению, чаще всего цели не определяются вообще или формулируются так: «Целью работы является создание системы...» Другими словами, целью создания системы является создание системы. О каких критериях достижения цели, критериях эффективности может быть речь?

Стратегические решения (второй уровень) направлены на долгосрочное планирование и формирование политики организации. Они отвечают на вопросы: «**Что** является итогом (результатом) деятельности, направленной на достижение цели?», «**Сколько** и каких ресурсов можно затратить на получение результатов?», «**Когда**, в какой срок, должны быть получены результаты?» На этом уровне необходима обширная информация, поступающая из самой организации и извне, а также средства документирования и сравнения вариантов и обоснованная нормативная база.

Какова же практика?

Решения о сроках и ресурсах принимаются волюнтаристически, без должного обоснования. Сравнение вариантов, ведущих к достижению цели, не производится. В результате все дольше и доро-

же, а в сухом остатке от деятельности отрицательный эффект, непонимание причин случившегося, дискредитация идей автоматизации, социальные конфликты, моральное развращение участников на всех уровнях и т. д. и т. п.

Одна из причин — отсутствие у решаемой проблемы хозяина, отсутствие серьезной (с ориентацией на доведение результата до уровня промышленного изделия) постановки вопроса о ее решении.

Предложение. Есть «команда», которая знает как и может* разработать систему для комплексной оценки трудозатрат и сроков разработки, прогнозирования необходимого количества специалистов различных специальностей и квалификации, выбора альтернативных стратегий разработки на всех этапах жизненного цикла программно-аппаратного изделия.

Для получения действующего макета необходимо полтора года, пять-шесть штатных единиц (группа, сектор). Инструментальными и объектными могут быть ЭВМ в диапазоне от Электроники-60 до СМ-1420.

Обращаться к автору статьи через редакцию журнала.

Примечание. Стандартный ответ, который мы получаем на это и аналогичные предложения: «Данная работа не входит в задачи нашей организации (ведомства)».

Тактические решения направлены на распределение ресурсов среднесрочного планирования и являются ответами на вопросы:

Как, т. е. каким (из известных и доступных) способом предполагается реализовать результат? Есть ли необходимость в параллельной проработке альтернативных вариантов? И т. д.

Кто конкретно выполняет задания, контролирует, координирует?

Где создаются отдельные компоненты и изделие в целом, размещаются люди и оборудование, производятся испытания, первоначальное внедрение и апробация?

Кому, от кого, куда, откуда? Определяются взаимосвязи, материальные и информационные потоки.

Чем? Определяются инструментальные средства деятельности.

Решения поцедурного (операционного) уровня связаны с повседневной текущей деятельностью организации и краткосрочным планированием.

Схема, рассмотренная нами, очень упрощена. В действительности процессы принятия решения итеративны и, кроме того, существует иерархическое расслоение уровней в соответствии с организационной структурой и иерархией компонентов изделия. Так, декомпозиция цели производится сверху вниз, начиная от руководителя верхнего уровня (рис. 3). После того как руководитель с помощью своего аппарата и экспертов определил цель, стратегию и тактику своего уровня, определяются цели руководителей, непосредственно ему подчиненных. Сумма их целей должна способствовать достижению цели верхнего уровня, но, кроме того, учитывает и личные цели руководителей.

Не исключено, что на этом этапе придется пересмотреть или цель верхнего уровня, или средства ее достижения (ресурсы, сроки), или состав руководителей нижнего уровня. И так до исполнителей самого нижнего уровня.

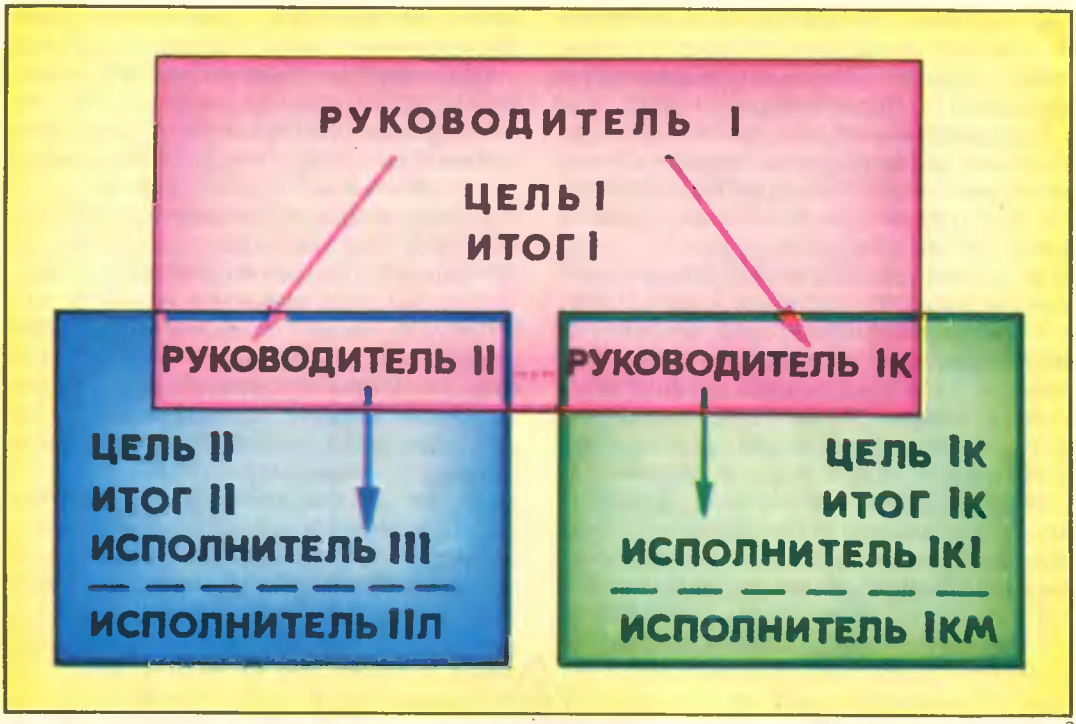
Как ни странно, но «непринятие» решений начинается с верхних уровней управления организацией, создающих и потребляющих автоматизированные системы. О принятии же решений в организации, которая управляет междуведомственной деятельностью по обеспечению систем-изделий, ничего сказать нельзя, так как такой организации нет.

Мы стали бояться четких формулировок. А вдруг не выполним?! Из-за недостатка информации нет внутренней уверенности в правильности выбранных путей. Из-за неумения обосновать ресурсные и временны́е ограничения соглашаемся за миллион в два года сделать то, на что мало десяти миллионов и пятилетки. Вместо того чтобы сконцентрировать ресурсы на важнейшем направлении и реальном результате, «размазываем» их, демонстрируя процесс функционирования.

Так изделия не делаются.

---

* Всю ответственность за истинность этого заявления несет автор.— *Прим. ред.*



3

Структура систем-изделий

Требования к изделию определяются в соответствии с интересами как изготовителей, так и потребителей, но главными являются интересы потребителей.

Первым требованием является общественная полезность (социальная значимость) изделия и интегральная эффективность при изготовлении и использовании.

Второе требование — изделие должно быть осуществимо в нужные сроки при имеющихся в наличии ресурсах.

Третье — возможность модификации изделия для устранения несоответствия с условиями заказчика, развития в разрешенных для текущей версии границах и адаптации к различным условиям использования.

Четвертое — необходимость осознания потребителями полезности изделия в их деятельности, обеспечиваемого средствами для своевременного информирования потенциальных пользователей о назначении изделия, его возможностях и выполняемых функциях, сфере использования и т. п.

Пятое — необходимость создания у потребителя уверенности в практической доступности для него изделия и возможности его освоения и использования в настоящем и будущем при определенных ограничениях на ресурсы и сроки освоения.

Шестое требование — необходимость предоставления потребителям изделия гарантий его надежности и качества.

Определение изделия. В соответствии с перечисленными требованиями доопределяем понятие ИЗДЕЛИЕ. Продукция является изделием, если она социально значима, определены конкретные потребители, свойства продукции (в частности, гарантии качества и надежности), критерии суммарной эффективности для потребителя и производителя. При этом система-изделие должна иметь все виды обеспечения, предусмотренные для нее.

Поскольку терминология в данной области еще не определена, предлагаем виды обеспечения систем, рассмотренные выше (см. рис. 1), обозначить как внутреннее обеспечение. Соответственно определим внешнее обеспечение, которое, собственно, и делает систему изделием (рис. 4).

Ситуация с внешним обеспечением напоминает айсберг. Верхушку, объектную систему (систему, поставленную на объекте автоматизации), видят все, а остальные восемь, как правило, ускользают от внимания и в лучшем случае реализуются фрагментарно из имеющихся под рукой универсальных средств в отрыве друг от друга.

В структуре внешнего обеспечения можно выделить три пары систем: объектная и обслуживания; разработки и сопровождения; производства и внедрения.

Их связывают системы освоения (информационного обслуживания изделия и обучения), поддержки, испытания.

Для каждой из систем внешнего обеспечения создается собственное внутреннее обеспечение в полном объеме (см. рис. 1). Таким образом, любая из систем внешнего обеспечения может быть даже более сложной, чем объектная система.

Объектная и обслуживающая системы. Основой изделия является объектная система, предназначенная для обеспечения его основных потребительских функций. Она ориентирована на специалистов автоматизируемой области (объекта автоматизации) и определяет специфику создаваемого изделия.

Система обслуживания является как бы продолжением объектной и ориентирована на административный и обслуживающий персонал, предметом деятельности которого являются все виды внутреннего обеспечения, поставленные заказчику для эксплуатации.

Если эта пара систем не разрабатывается совместно, то реализовать систему обслуживания в процессе эксплуатации практически невозможно, что мы и

видим на практике. Руководству заказчика невозможно объяснить, почему, заплатив большие деньги за автоматизированную систему, оно должно создавать не только подразделения электронщиков, механиков, системных программистов, но совершенно непонятные группы Администратора базы данных, Архива машинных носителей и информации, Администратора системы, Связи и т. д. Более того, для этих групп еще надо разрабатывать дополнительные средства автоматизации. При этом возникает необходимость в перестройке купленной системы.

В итоге система (даже хорошая) работает кое-как. Возникают претензии к подразделению обслуживания, неукомплектованному и не обеспеченному необходимыми средствами. Обслуживающий персонал (даже грамотный), не имея возможности вскрыть действительные причины непорядков, начинает валить все грехи (в том числе и свои) на разработчиков и производителей, причем в системе действительно сложно разграничить ошибки обслуживания, разработки и производства.

Системы разработки и сопровождения относятся к классу САПР и должны выполняться с соблюдением требований «Общеотраслевых руководящих методических материалов по созданию систем автоматизированного проектирования» (ОРММ САПР) ГКНТ СССР.

Положение в этой области, как ни странно, очень плохое. Если разработчики программ и аппаратуры еще *могут* быть кое-как обеспечены инструментальными средствами, то разработчики информационного, организационно-правового, лингвинистического и эргономического обеспечения должны работать «вручную»; правда, *могут* — еще не значит *обеспечены*. Разработчикам приходится собирать инструментальные средства кустарным способом при ограниченных ресурсах в сроки, отпущенные для разработки.

Система сопровождения (модификации с целью устранения ошибок или развития изделия) должна обладать еще более развитыми средствами моделирования, документирования, хранения и представления информации, чем система разработки. Эти средства предоставляются службам сопровождения на все время жизненного цикла изделия. Большие затраты в сочетании с организационными и методологическими проблемами, которые мы рассмотрим в разделе «Функции и службы жизнеобеспечения изделия», являются причиной того, что даже профессиональные идеологи обеспечения изделий надеются в основном на талант и необыкновенные личные качества сопровождающих специалистов.

Системы производства и внедрения. Возможность создания изделия зависит от уровня технологии производства. Это должно быть учтено в процессе разработки. В свою очередь, производство конкретного изделия должно быть оснащено средствами изготовления и настройки, учитывающими его специфику.

Внедрение является продолжением производства изделия на площадке заказчика и должно быть оснащено инструментальными средствами монтажа, наладки, первичного наполнения информацией и сдачи в опытную эксплуатацию. Заказчик должен получать систему не по описи комплектующих, а в работоспособном состоянии. Причем должна быть продемонстрирована работоспособность не отдельных устройств, а системы в целом.

В настоящее время существует производство устройств, плат, ЭВМ. О производстве программно-аппаратных, информационных компонентов и систем в целом приходится только мечтать. На площадке потребителя монтируются и налаживаются только ЭВМ, и то на низком профессиональном уровне.

Система испытания изделия является инструментом обеспечения качества и надежности. Она представляет собой обширный комплекс средств, используемых на всех стадиях жизненного цикла изделия для разного рода испытаний. Более того, принимая во внимание, что изделие включает в себя другие изделия, можно считать ее многоуровневой иерархической системой.

Испытания должны проводиться на разных уровнях — система в целом, подсистемы, отдельные виды обеспечения, программно-аппаратные комплексы, устройства, микросхемы и т. д.

Испытания могут быть демонстрационные, аттестационные, функциональные, надежностные, контрольные, пусковые, эксплуатационные и т. п.

Специалисты, анализирующие технологию обеспечения жизненного цикла систем-изделий, считают, что эта область деятельности еще мало изучена и в зарубежных фирмах является самой секретной частью разработки. Действительно, хотя для многих видов испытаний уже имеются инструментальные средства, в целом эта система еще ожидает своих исследователей и разработчиков.

Система поддержки изделия представляет собой большой класс инструментальных средств, используемых соответствующей службой для обеспечения организационно-технической подготовки, опытной эксплуатации, сдачи в эксплуатацию в процессе внедрения изделия. На стадии эксплуатации система используется для обеспечения внесения изменений в действующее изделие; восстановления изделия после аварийных ситуаций; выявления, регистрации и доведения до службы сопровождения заявок на расширение возможностей и устранения выявленных в процессе эксплуатации ошибок в изделии.

Системы информационного обслуживания и обучения. Это, собственно, комплекс систем. Системы информационного обслуживания и обучения взаимно дополняют друг друга. Пользователями их могут быть все специалисты, соприкасающиеся с изделием.

Система информационного обслуживания предназначена для обработки, регистрации, хранения и предоставления информации и документации пользователям, разработчикам, изготовителям и другим службам на всех стадиях жизненного цикла. Ее можно трактовать как часть общенациональной информационной сети, которая контролирует информацию внутри среды обеспечения изделия, перекачивает дополнительную информацию извне, предоставляет информацию по изделию внешним коллективам.

Система обучения предназначена для использования в службах обеспечения изделия, а также в институтах повышения квалификации, ПТУ, техникумах, вузах. Она должна охватывать все виды деятельности и всех специалистов по обслуживаемому изделию в режимах обучения, самоподготовки, повышения квалификации, переподготовки и освоения.

Компоненты систем-изделий

Сложность реализации систем как изделий заключается еще и в том, что они состоят из функционально законченных компонентов разных уровней сложности, которые поставляются различными организациями. Это могут быть подсистемы, программно-аппаратные комплексы, информационные компоненты (например, нормативная информация, правовые документы) и т. д.

Существуют компоненты более низкого уровня, используемые для комплектации изделия,— устройства, микросхемы и т. д. При создании изделия необходимо, чтобы все его комплектующие также были изделиями. Однако следует помнить, что сумма комплектующих изделий не становится изделием автоматически. Для этого нужны немалые дополнительные усилия и ресурсы.

Вопрос о декомпозиции на компоненты-изделия и организации их жизненного цикла с последующей интеграцией требует отдельного анализа. Здесь же мы в качестве примера разберем, можно ли выделять программу-изделие как самостоятельный компонент.

Дело в том, что программисты потратили много сил, чтобы доказать самостоятельную ценность программ и необходимость создавать индустрию программного обеспечения. В результате сложилась парадоксальная ситуация — программы приобрели самоценность в глазах не только руководителей различных рангов, но и самих программистов. В пылу сражений за статус программного обеспечения почти забыли, что программы и аппаратура составляют единое функциональное целое. В ГОСТах предусмотрена необходимость совместимости программ и технических средств. Более того, требуется испытывать программы на различных потенциально возможных конфигураци-

ях технических средств. Однако при разработке программ об этом довольно часто забывают и даже умудряются не указывать в документации, для каких версий комплекса технических средств гарантирована адекватность работы программы.

В действительности минимальной единицей разработки систем следует считать программно-аппаратный комплекс и соответственно программно-аппаратное изделие. Причем даже если в существующий программно-аппаратный комплекс встраивается хотя бы одна программа, необходимо, как правило, считать, что появился новый комплекс, и соответственно проводить испытания и дорабатывать все присущие ему компоненты как для нового изделия.

Соотношение этапов, функций и служб обеспечения жизненного цикла изделий

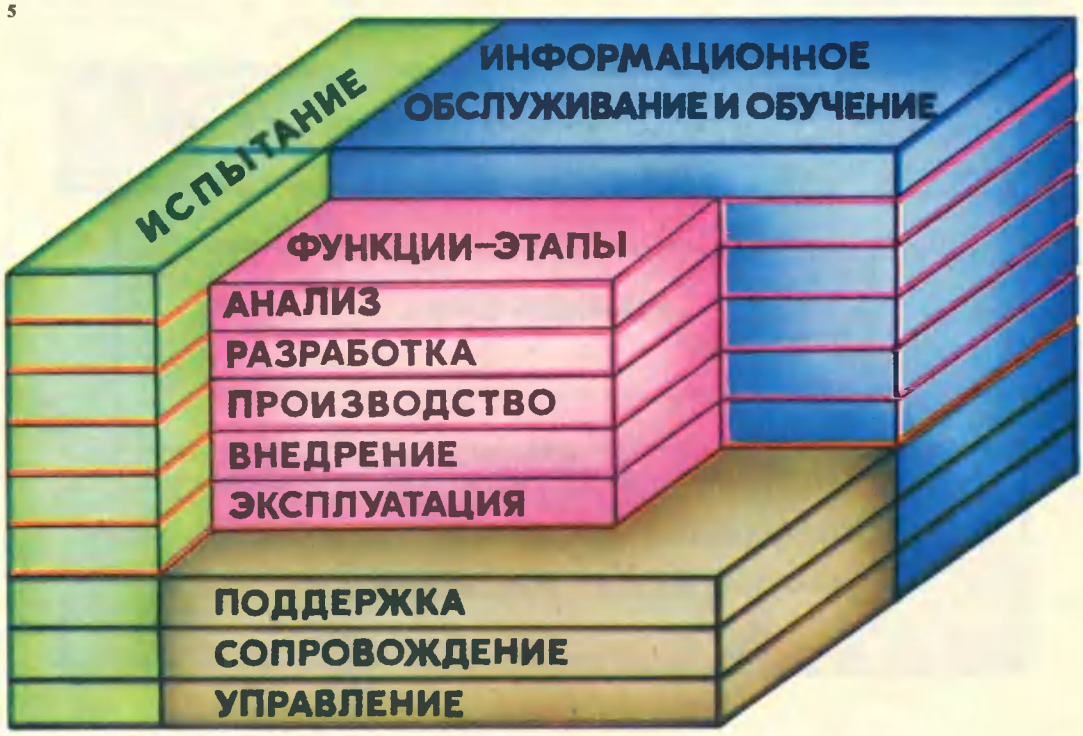
На рис. 5 показаны функции, выполняемые в процессе обеспечения жизненного цикла изделий. Условно их можно разделить на четыре группы. Фундамент составляют функции, которые действуют на протяжении всего цикла и в отдельные этапы не выделяются. Это управление, сопровождение и поддержка.

Вторую группу составляют функции, которые действуют на соответствующих этапах: анализ, разработка, производство, внедрение и эксплуатация. Третью группу составляет испытание — она предназначена для испытания результатов деятельности всех функций на всех этапах. Четвертая группа — информационное обслуживание и обучение всех специалистов, вовлеченных в обеспечение изделия, на всех этапах.

Для обеспечения функций создаются соответствующие службы. Существенной особенностью служб обеспечения функций в жизненном цикле является то, что они комплектуются разными ведомствами (например, на правах междуведомственных целевых комплексных коллективов). Службы, созданные для обеспечения комплектующих компонентов (изделий более низкого уровня), технологически связаны с соответствующими службами изделия. Их деятельность по этому изделию должна координироваться общим для них органом управления.

Центральными службами обеспечения

5



следует считать управление, сопровождение и поддержку. Они создаются в самом начале жизненного цикла для управления и координации, определения требований и технической политики для изделия и его компонентов (рис. 6). Поэтому их статус должен быть достаточно высоким и подкреплен соответствующими организационно-правовыми документами. Именно эти службы определяют — быть ли изделию, каким оно должно быть, когда оно снимается с сопровождения и поддержки. Более того, если службы не созданы, то о разработке изделия не может быть и речи.

На рис. 7 показана иерархия служб сопровождения и поддержки, соответствующая иерархии изделий. Представлено четыре уровня изделий. Нижний уровень, комплектующих, включает в себя устройства, программы. Компоненты систем (программно-аппаратные комплексы, подсистемы) включают в себя изделия предыдущего уровня и входят в качестве элементов в верхний уровень, изделием которого является система. Если распространить концепцию изделия на автоматизированное учреждение, то системы, в свою очередь, войдут в него как элементы.

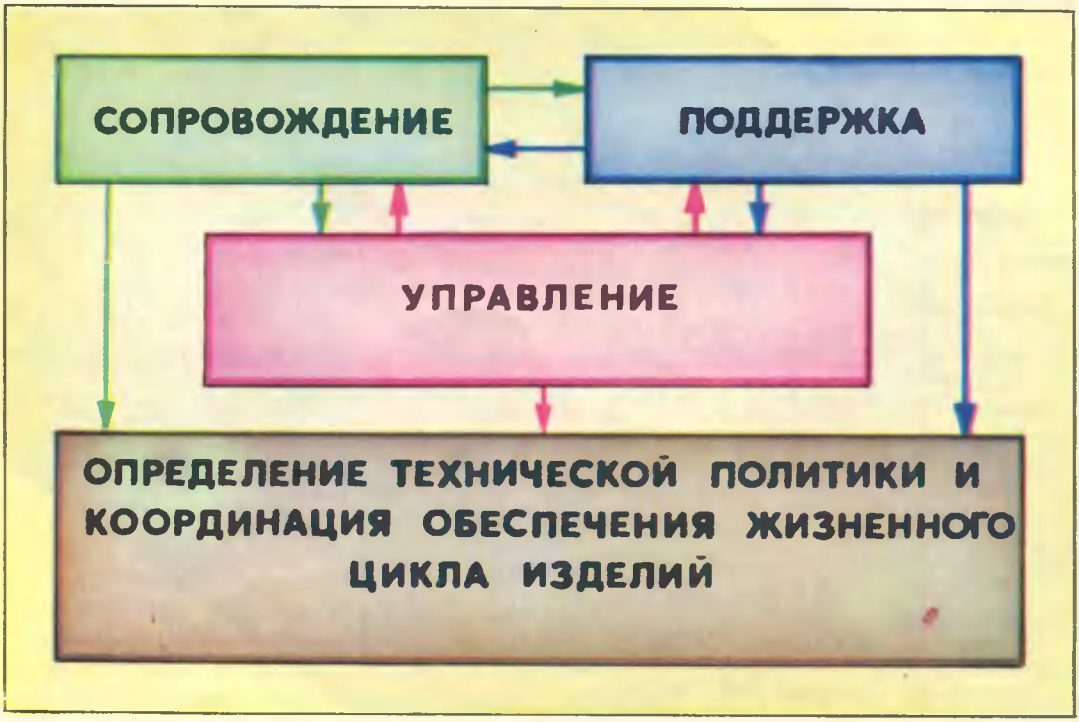
Таким образом, потребители изделия находятся на следующем, верхнем уровне. Примем правило, что служба поддержки (защиты интересов потребителя) создается потребителем, а служба сопровождения (гарантия качества) создается производителем.

Службы поддержки комплектующих изделий организуют создатели компонентов систем. Поддержку компонентов систем обеспечивают создатели систем.

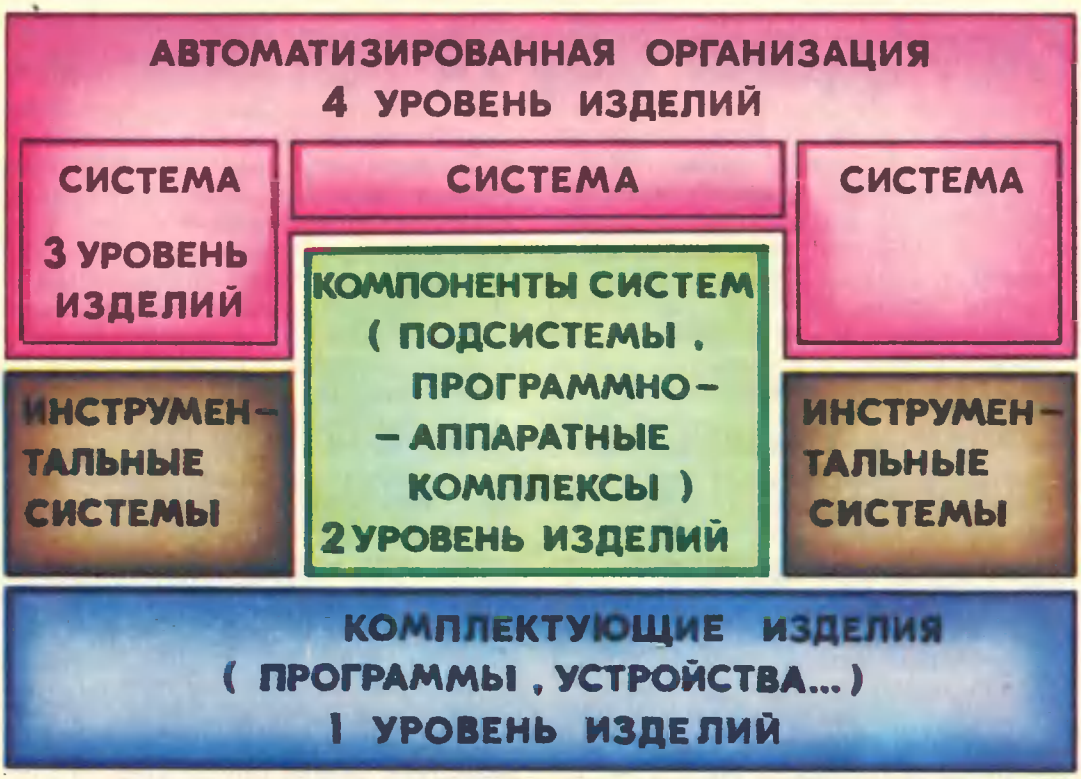
Поддержка систем может осуществляться как средствами самой системы, так и дополнительными средствами автоматизированного учреждения.

Службы второй группы создаются по инициативе служб первой группы в соответствии с программой обеспечения изделия. При этом должна быть предусмотрена возможность создания нескольких групп разработки или экспертизы, которым поручена одна и та же задача, если доказана необходимость реализации альтернативных вариантов.

Службы испытания результатов деятельности других служб создаются в та-



6



кие сроки, чтобы они успели разработать методики и инструментальные средства для испытаний. Кроме того, они должны определить свои требования к испытываемым результатам и контролировать их исполнение.

Службы информационного обслуживания и обучения создаются как элементы соответствующей сети служб, которая в целом не привязана к какому-либо одному изделию.

Окончание следует.

Уважаемые читатели!

Наш журнал распространяется только по подписке. К сожалению, редакция не может высылать читателям номера журнала наложенным платежом. Не забудьте своевременно продлить подписку на 1987 г.!

А. КУЗНЕЦОВ, Д. СМЕКАЛИН

Проверка и оценка знаний и умений школьников по основам информатики и вычислительной техники*

Одним из основных направлений повышения эффективности учебного процесса по курсу основ информатики и вычислительной техники (как и любого учебного предмета) является совершенствование проверки и оценки результатов обучения школьников. Проверочно-оценочная деятельность учителя — неотъемлемая часть всей его педагогической работы, важный фактор улучшения качества обучения.

Нормативный документ, на который должен ориентироваться учитель при проверке и оценке знаний школьников,— программа курса «Основы информатики и вычислительной техники», в которой выделен специальный раздел «Требования к знаниям и умениям учащихся». Важно, чтобы контроль и оценка знаний учащихся отвечали общедидактическим требованиям и выполняли учетную, контрольно-корректирующую, обучающую, воспитательную функции.

Учетная функция контроля проявляется в систематической фиксации результатов обучения, что позволяет учителю судить об успеваемости каждого ученика, его достижениях и недочетах в учебной работе.

Контрольно-корректирующая функция обеспечивает обратную связь «учитель—ученик», необходимую для внесения учителем коррективов в методику обучения, некоторого перераспределения учебного времени между различными вопросами темы, вызываемых недочетами в знаниях школьников, уровнем подготовки класса. Особенно важна эта функция контроля при обучении информатике именно сейчас, когда и сам предмет, и методика обучения ему находятся в стадии становления.

Обучающая функция контроля проявляется в том, что в процессе проверки состояния знаний, умений и навыков школьников происходит повторение материала, учитель акцентирует внимание класса на главных вопросах и важнейших мировоззренческих идеях курса, указывает на типичные ошибки, что способствует углублению знаний учащихся.

Воспитательная функция контроля и оценки подразумевает стимулирование учащихся к дальнейшей учебной работе, углублению своих знаний. Оценивая работу ученика, учитель не просто констатирует состояние знаний, умений и навыков школьника, но и направляет его в учебной работе, дает дополнительную мотивацию в познавательной деятельности.

Анализ первого года работы учителей информатики показывает, что в практике проверки и оценки знаний есть существенные недостатки, снижающие эффективность учебно-воспитательного процесса. Наблюдается формаль-

---

* Методическое письмо. Одобрено отделением дидактики и частных методик АПН СССР

ное применение методов и средств проверки, в ряде случаев субъективизм в оценке знаний учащихся. Часть учителей преуменьшает обучающую роль проверки, пренебрегает содержательными критериями оценки, не привлекает учащихся к оценочной деятельности, не развивает у них умений самоконтроля. Некоторые учителя недооценивают значение учетно-контрольной функции проверки, недостаточно ориентируют обучение на его конечные результаты, не всегда учитывают программные требования к результатам обучения, завышают или занижают эти требования.

Имеют место случаи, когда учитель требует от учащихся обязательного усвоения внепрограммного материала, обращается в процессе проверки к несущественным фактам и деталям, не акцентируя внимание на основном, главном в учебном материале. Последнее, например, часто имеет место, когда некоторые учителя уже в начале курса пытаются использовать в качестве средства описания алгоритмов язык программирования, перегружая школьников формальными правилами программ на этом языке, вместо того чтобы уделить главное внимание самому понятию алгоритма и свойствам алгоритмов.

Несмотря на то что программа и методическое пособие для преподавателей информатики ориентирует учителя на использование разнообразных форм, средств и методов проверки, часть учителей неоправданно ограничивается применением лишь некоторых из них, что существенно снижает возможности выявления результатов обучения, реализации основных функций проверки.

Например, многие учителя для контроля знаний ограничиваются устным опросом школьников, в процессе которого лишь пересказывается текст учебника. Явно недостаточно проверяется умение учащихся решать задачи. Некоторые учителя практически не используют при проверках задания на исполнение алгоритмов, несмотря на то что подобные упражнения наиболее явно демонстрируют понимание учащимися темы и развивают у них навыки самоконтроля.

Изучение опыта работы школ показало, что ряд учителей еще недостаточно хорошо освоили ведущие методические идеи программы курса, недооценивает важности ее раздела «Требования к знаниям и умениям учащихся».

В соответствии с требованиями программы учителю необходимо ориентировать преподавание, контроль и оценку на достижение высоких конечных результатов во всех видах учебной деятельности. При этом особое значение должно придаваться освоению специальных умений, отражающих особенности этого учебного предмета, а также общеучебных умений, развивающих умственные способности учащихся, способствующих формированию навыков самостоятельно учиться, работать с различными источниками знаний, рационально организовывать свою учебную деятельность. На этой основе определяются конкретные цели и объекты проверки учебной подготовки школьников, виды и формы ее контроля, критерии ее оценки.

Следует различать назначение отдельных видов контроля в учебном процессе: формирующего (текущего), зачетно-тематического, итогового.

По результатам текущей проверки учитель должен решать вопрос, готовы ли школьники полноценно усваивать последующий учебный материал. Отличительной особенностью формирующего контроля является его проведение на всех этапах изучения темы или раздела: ознакомления с учебным материалом, формирования и развития знаний и умений, их закрепления и углубления. Из этого следует, что в процессе текущего контроля от учащихся можно требовать знания только на том познавательном уровне (воспроизведения и применения знаний в стандартной ситуации, в измененной ситуации или в новой ситуации), какой предусматривается определенным этапом овладения учебным материалом. Завышение требований к учебной подготовке, когда знания и умения еще не сформировались, или, наоборот, занижение требований в конце изучения темы — одна из основных причин необоснованной оценки результатов обучения.

Для эффективного применения формирующего контроля необходимо применять разнообразные формы и средства проверки в их рациональном сочетании: фронтальные и индивидуальные, устные и письменные, рассчитанные на весь урок или его часть.

Опыт передовых учителей свидетельствует, что повышению эффективности текущего контроля служит организация учета результатов учебной деятельности школьников. Для подведения итогов обучения учителю нужно знать, за усвоение каких материалов и вида учебной деятельности выставлена текущая отметка. Такой учет может быть обеспечен, если в классном журнале конкретно и достаточно полно фиксируются тема урока и содержание домашних заданий. Представляется целесообразным выделять специальные колонки в классном журнале для оценки результатов различной учебной деятельности, выполнения разных типов проверочных заданий.

Для повышения объективности оценки необходимо проводить контроль по итогам изучения отдельных тем и разделов учебного курса, когда знания в основном сформированы, систематизированы. Цель такого контроля — выявление уровня овладения основным содержанием темы каждым учащимся и классом в целом. В содержание контроля должны войти основные вопросы темы, которые отбираются в соответствии с требованиями к результатам обучения и зафиксированы в программе.

Итоговая проверка усвоения учебного материала должна быть спланирована таким образом, чтобы можно было проконтролировать усвоение школьниками основных элементов программного материала. Знания по итогам изучения темы могут быть оценены положительно, если учащиеся овладели всеми основными элементами программного материала.

Тематический контроль может проводиться как в форме письменной контрольной работы, так и в форме зачетных занятий по пройденной теме. При проведении тематического контроля часть заданий должна соответствовать деятельности по образцу, а часть — деятельности в измененной и новой ситуациях, что предоставит каждому учащемуся возможность полностью проявить уровень своей подготовки по теме.

Программой курса «Основы информатики и вычислительной техники» выделены шесть основных тем:

алгоритмы;

алгоритмический язык;

алгоритмы работы с величинами;

построение алгоритмов для решения задач;

принципы устройства и работы ЭВМ;

знакомство с программированием;

роль ЭВМ в современном обществе.

Специфическая особенность курса информатики — его направленность на подготовку учащихся к последующему использованию ЭВМ в практической деятельности, выработки у них определенных операциональных умений и навыков. Это накладывает отпечаток и на проверку и оценку знаний школьников — объектом контроля становится прежде всего их готовность к различного вида учебной деятельности, сформированность определенных умений и навыков.

В результате изучения темы «Алгоритмы. Алгоритмический язык» школьники должны освоить правила записи простейших алгоритмов с использованием команд ветвления и повторения. При этом они должны понимать сущность алгоритмов, возможность их формального исполнения, сознавать преимущество, вытекающее из стандартизации формы записи алгоритмов. На проверку этих знаний и умений школьников ориентирован текущий контроль по данной теме.

В качестве проверочных заданий ученикам следует использовать не только запись на алгоритмическом языке известных им алгоритмов из курсов математики и физики (чем, к сожалению, часто ограничиваются), но и построение новых простых алгоритмов. Должны быть проверены умения школьников исправить команды алгоритма в расчете на конкретного исполнителя, построить алгоритм с использованием фиксированной системы команд.

Очень информативны задания на исполнение алгоритмов. При этом ученику может быть предложен готовый алгоритм, построенный учителем или

другим учащимся. Последнее более предпочтительно, так как помимо непосредственно проверочной ценности способствует повышению интереса учащихся к выполняемой ими работе и содействует развитию у них навыков самоконтроля.

Другим важным объектом контроля по теме «Алгоритм. Алгоритмический язык» является осознанное использование учащимися команд ветвления и повторения. При этом особое внимание рекомендуется уделить проверке понимания школьниками сущности команды повторения, которая, как показывает опыт, вызывает у многих большие затруднения. Для этой цели могут использоваться задания следующего вида:

построить простейший алгоритм с использованием команды повторения;

определить, какое количество раз будет выполнена серия команд, входящая в команду повторения, в уже готовом алгоритме;

записать условие в уже готовую команду повторения;

заменить серию повторяющихся команд на команду повторения;

привести примеры простейших повторяющихся процессов (бытовых, производственных, учебных);

предложить условие задачи (из школьных курсов или бытовую), в алгоритме решения которой удобно использовать команду повторения.

Наконец, именно в первой теме курса закладывается используемая в дальнейшем терминология, система служебных слов и правил записи алгоритма. Усвоение их учащимися также должно быть объектом контроля со стороны учителя. Необходимо следить за правильным использованием выражений «исполнение алгоритма», «выполнение команды», «соблюдение условия» и других, проверять правильность использования и записи служебных слов, аккуратного оформления алгоритмов.

Тема «Алгоритмы работы с величинами» — центральная в курсе IX класса. В ней учащиеся знакомятся с понятием величины, как средством обозначения различного рода информации при записи алгоритмов, правилами исполнения алгоритмов с использованием таблицы значений, использованием отношений для записи условий (в том числе и составных условий), вспомогательными алгоритмами и некоторыми особенностями построения и исполнения алгоритмов с использованием табличных величин и вспомогательных алгоритмов. Основная цель этой темы — привить школьникам навыки алгоритмизации, научить их строить разнообразные алгоритмы с использованием достаточно широкого спектра средств, предоставляемых алгоритмическим языком.

Важнейшим методическим средством текущей проверки состояния знаний учащихся при изучении этой темы является исполнение школьниками алгоритмов. Исполнение учеником алгоритма с использованием таблицы значений наглядно демонстрирует понимание им проверяемого материала. Это относится ко всем параграфам и понятиям темы. Порядок записи шагов алгоритма в таблице значений показывает, насколько школьник знает правила выполнения команд алгоритмического языка. Выделение в таблице колонок для аргументов, результатов и промежуточных величин, запись в них значений, соответствующих типам указанных величин, позволяет учителю проконтролировать уровень освоения учеником понятий величины и их типов, умение правильно использовать информацию, содержащуюся в заголовке алгоритма, при его исполнении. Колонка «проверка условий» дает возможность определить знание школьником правил проверки соблюдения условий.

Одновременно накопление учащимися опыта исполнения алгоритмов развивает у них навыки самоконтроля, способствует более полному и глубокому освоению изучаемого материала.

Вместе с тем нужно отметить, что в отличие от алгоритмического языка, имеющего четкие правила, форма представления таблицы значений может быть достаточно произвольной и модифицироваться учителем и школьником. Поэтому не следует требовать от школьников чрезмерно жесткого соблюдения всех правил графического оформления

таблиц, предложенных в учебнике. В таблице значений для учителя важно ее содержание. Наличие у учащихся графических погрешностей при правильной фиксации шагов алгоритма и значений величин не должно влиять на снижение отметки. Например, можно позволять учащимся использовать таблицу значений не только для записи окончательного ответа, но и для проведения промежуточных вычислений. Так, выполняя команду «$x:=2\cdot a+4{,}3$», где $a=2{,}1$, ученик может записать в соответствующей колонке: «$2\cdot 2{,}1+4{,}3=$ $=4{,}2+4{,}3=8{,}5$». Важно только, чтобы он понимал, что окончательное значение x — именно 8,5. При проверке составного условия «$a>2$ и $a<10$» для того же значения a в колонке «проверка условий» не будет ошибочной запись вида «$2{,}1>2$ и $2{,}1<10$= да и да=да» и т. п.

Использование исполнения алгоритмов для текущего контроля вовсе не означает обязательного задания учащимся исполнить алгоритм целиком. Возможны отдельные задания на выделение аргументов, результатов и промежуточных величин, указание типов величин, проверку условий, выполнение отдельных команд. В этом случае не следует требовать от школьников построения всей таблицы значений, достаточно лишь указания ее необходимых фрагментов.

Как показал опыт первого года обучения, наибольшие затруднения у учащихся вызывают понятия литерных величин, табличных величин и использование вспомогательных алгоритмов. Этим понятиям, прежде всего с точки зрения реализации обучающих функций проверки, и должно быть уделено особое внимание при текущем контроле знаний школьников.

При отработке понятия литерной величины учащиеся должны четко понимать разное отношение к значениям этих величин со стороны составителя алгоритма и его формального исполнителя. Для составителя алгоритма значение литерной величины имеет важную содержательную информацию, в то время как для исполнителя это лишь набор символов. При контроле этого положения полезны задания на исполнение алгоритма, в которых литерные величины получают значения бессмысленных сочетаний знаков, неверно написанных слов и т. п. Они помогают проверить понимание школьниками формальности работы исполнителя с литерными величинами. С другой стороны, при построении алгоритмов целесообразны задания на внесение в уже готовые тексты алгоритмов команд, содержащих различные комментарии результатов исполнения (аналогично примерам 9.1 и 9.2 из ч. I учебника).

Применение в алгоритмическом языке табличных величин неразрывно связано с использованием команды повторения. Для контроля усвоения этих понятий рекомендуется предлагать задания типа:

заменить серию команд присваивания значений однотипным величинам (числовым или литерным) командой повторения, присваивающей значения элементам табличной величины;

составить описание (указать границы) таблиц, обрабатываемых с помощью команды повторения, в соответствии с условием команды повторения;

подставить условие в команду повторения, изменяющую значение элементов таблицы, ориентируясь на описание таблицы, и т. п.

Для контроля навыков использования вспомогательных алгоритмов методически оправданным является построение алгоритмов с использованием алгоритмов из библиотеки. Учащиеся должны понимать, что вспомогательные алгоритмы (библиотечные) для исполнителя уже известны, их ему описывать не надо. При проверке рекомендуется предлагать школьникам задания построить алгоритм из нескольких команд, но содержащий вызов вспомогательного алгоритма.

Отметим, что методическое пособие для преподавателей курса рекомендует проведение по теме «Алгоритмы работы с величинами» не менее двух самостоятельных работ и итоговую контрольную работу по всему материалу темы.

Следующая тема курса информатики IX класса — «Построение алгоритмов

для решения задач» — имеет четкую прикладную направленность. Ее задача — отработать и закрепить у школьников навыки использования средств алгоритмического языка для построения алгоритмов, а также познакомить их с некоторыми методами построения алгоритмов для решения практических задач.

Как показал опыт, учебный материал, предложенный в § 5 «Этапы решения задач», не вызывает больших затруднений у школьников. Сами этапы являются достаточно естественными и знакомы учащимся из опыта решения задач из других школьных курсов. Поэтому проверку знаний учениками материала этого параграфа рекомендуется проводить одновременно при решении ими содержательных задач. Разбирая решения подобных задач, целесообразно предложить учащимся обосновать порядок своих рассуждений при выполнении всего задания.

Наиболее содержательными разделами темы являются алгоритмы работы с табличными величинами, решения задач из курса математики и работы с графической информацией. Успешность освоения школьниками этого материала должна быть хорошо проконтролирована учителем.

Основной формой текущего контроля по теме служат задания на построение и исполнение алгоритмов. По материалу параграфа «Алгоритмы для работы с табличными величинами», развивающему материал предыдущей темы и показывающему некоторые конкретные случаи использования таблиц в алгоритмах и способы их обработки, должны тщательным образом формироваться и контролироваться навыки школьников по определению границ таблицы, заданию значения параметрам команды повторения, правильный выбор номера элемента таблицы. Рекомендуется также использовать при проверке задания на исполнение алгоритмов работы с табличными величинами, предлагать учащимся записать начальный и конечный вид небольших таблиц, значения элементов которых меняются при исполнении алгоритма.

Материал параграфов по решению задач из курса других учебных предметов призван показать учащимся широту применения алгоритмов, возможность использования вычислительных методов вместо аналитических.

Последнее положение является чрезвычайно важным положением информатики и должно быть хорошо освоено учащимися. Для контроля рекомендуется использовать задания типа «составить вычислительный алгоритм, с помощью которого можно было бы решить целый ряд различных задач». Например, с помощью алгоритма «уточнение корня» может быть решен целый ряд уравнений самого различного вида.

Особенное внимание в этой теме должно быть уделено совместному использованию известных учащимся методов при решении задач. Следует использовать проверочные задания на построение алгоритмов с использованием вспомогательных алгоритмов из библиотеки или построенных ими ранее при изучении различных тем курса. Например, могут быть предложены задания на нахождение корней функции и занесение их в таблицу упорядочения значений многочлена для разных по возрастанию с помощью алгоритмов «Схема Горнера» и «упорядочение» и т. п. После изучения параграфа «Алгоритмы работы с графической информацией» учащимся могут быть предложены задания построения алгоритмов вычерчивания графиков простейших математических функций.

Методическим руководством для преподавателей при контроле знаний по теме «Построение алгоритмов для решения задач» предусмотрено проведение не менее трех самостоятельных работ и итоговой контрольной работы, проверяющей знания учащихся на конец первого года обучения в полном объеме.

Проверка и оценка знаний и умений в X классе по информатике основывается на тех же методических принципах и критериях, что и в IX классе.

Главная цель темы «Устройство ЭВМ» в X классе — ознакомить школьников с основными особенностями нового формального исполнителя

алгоритмов — ЭВМ. Изучив ее, учащиеся должны понимать, почему и каким образом может осуществить компьютер автоматическое исполнение алгоритмов работы с величинами, с которыми школьники познакомились в IX классе.

Центральным материалом этой темы является описание основного алгоритма работы процессора, ему и должно быть уделено особое внимание при текущем контроле состояния знаний учащихся. Основным методическим средством учителя при проверке знаний школьников этого материала остается, как и в IX классе, исполнение учащимися алгоритмов. Предлагаемая форма таблиц значений для исполнения алгоритмов, представленных в виде машинных команд, приведена в учебнике (ч. II). Особое внимание учитель должен уделить контролю за правильным указанием в таблице адресов команд и значений регистра «счетчик команд». При разборе алгоритмов (программ) рекомендуется в качестве текущей проверки предложить учащимся указать, как работает основной алгоритм процессора при выполнении той или иной команды.

Отметим, что выучивание школьниками всего набора машинных команд не является обязательным. Поэтому при проверке знаний учащихся можно разрешать им пользоваться справочными таблицами перевода основных конструкций алгоритмического языка в машинные команды. Последнее относится и к самостоятельным, и к контрольным работам.

При контроле знаний учащихся по материалу параграфа «Физические принципы устройства ЭВМ» можно рекомендовать задания: указать значения электрических сигналов на входах и выходах электронных устройств простейших логических схем.

В качестве итоговой проверки по теме методическим пособием для преподавателей рекомендуется проведение контрольной работы.

Тема «Знакомство с программированием» разделена в пробном учебном пособии на две подтемы:

знакомство с новыми конструкциями алгоритмического языка, показывающими богатство средств записи алгоритмов в реальных языках программирования;

знакомство с основными правилами и командами языков программирования Рапира и Бейсик (выбор языка для изучения школьниками предоставляется учителю).

Новые команды алгоритмического языка (выбора, повторения с параметром, вызова вспомогательных алгоритмов вычислений значений функции, команды работы с литерными величинами) дополняют изученный школьниками материал в IX классе (команды ветвления, повторения, вызова вспомогательных алгоритмов, присваивания). Поэтому рекомендации по осуществлению текущего контроля знаний учащихся остаются теми же, прежде всего — исполнение школьниками алгоритмов.

Методическое пособие для преподавателей рекомендует проведение по этой теме самостоятельной и контрольной работы.

Материал учебника, посвященный языкам программирования, не рассчитан на получение школьниками глубокого знания этих языков. Особое внимание в проверочной деятельности учителя должно быть обращено на понимание школьниками особенностей языка программирования, необходимости усиления стандарта точности при записи команд и выражений. В качестве контрольных заданий рекомендуется перевод фрагментов алгоритмов в соответствующие фрагменты программ в письменных работах и объяснение хода исполнения и назначения отдельных команд в фрагментах программ при устных опросах.

Отметим, что методическое пособие для преподавателей рекомендует провести итоговую проверку по этому материалу в форме письменной контрольной работы.

Тема «Роль ЭВМ в современном обществе» содержит в основном лекционный материал, рассчитанный на расширение кругозора школьников по вопросам истории вычислительной техники и областям ее применения.

Ведущей формой контроля знаний

по этой теме является беседа учителя со школьниками. В конце изучения материала можно рекомендовать проведение зачетного занятия. Другой формой аттестации знаний учащихся может быть подготовка школьниками рефератов по вопросам темы.

Необходимый элемент контроля знаний учащихся — оценка его результатов учителем. Важнейшим условием, способствующим повышению объективности отметки, должна стать четкая ориентация при ее выставлении на содержательные критерии оценки. Такая ориентация служит единообразному применению шкалы отметок, достижению единства в оценивании знаний школьников.

При оценке знаний необходимо учитывать основные качественные характеристики овладения учебным материалом: имеющиеся у учащихся фактические знания и умения, их полноту, прочность, умение применять на практике в различных ситуациях, владение терминологией и специфическими способами обозначения и записи.

Кроме того, при оценке устных ответов следует учитывать умение школьников использовать помощь учителя для исправления ошибок и углубления знаний.

Результат оценки зависит также от наличия и характера погрешностей, допущенных при устном ответе или в письменной работе. Среди погрешностей можно выделить ошибки, недочеты и мелкие погрешности.

Погрешность считается ошибкой, если она свидетельствует о том, что ученик не овладел основными знаниями и умениями и их применением.

К недочетам относятся погрешности, свидетельствующие о недостаточно прочном усвоении основных знаний и умений или отсутствии знаний, которые в соответствии с программой не считаются основными. Недочетом также считается погрешность, которая могли бы расцениваться как ошибка, но которая допущена в одних случаях и не допущена в других аналогичных случаях. К недочетам относятся погрешности, объясняемые рассеянностью или недосмотром, небрежная запись, небрежное оформление алгоритмов.

К мелким погрешностям относятся погрешности в устной и письменной речи, не искажающие смысла ответа или решения, случайные описки и т. п.

В каждом отдельном случае вопрос об отнесении погрешности к ошибкам, недочетам или мелким погрешностям решается учителем в соответствии с требованиями к усвоению материала на данном этапе обучения.

К ошибкам, например, относятся: неправильное использование служебных слов алгоритмического языка; неверное указание аргументов и результатов; присваивание величине одного типа значения другого типа; нарушение порядка выполнения команд при исполнении алгоритма; несоответствие команды вызова вспомогательного алгоритма описанию вспомогательного алгоритма и т. п.

Примеры недочетов: пропуск или неправильная запись служебного слова алгоритмического языка; описаны не все промежуточные величины; случайные вычислительные погрешности при проверке условий составных команд; небрежное оформление записи алгоритма и т. п.

Если одна и та же ошибка (недочет) встречается несколько раз, то это рассматривается как одна ошибка (один недочет). Зачеркивания и исправления в письменных работах свидетельствуют о поисках решения, что считать ошибкой не следует.

Задание для устного опроса или письменной работы считается выполненным безупречно, если содержание ответа точно соответствует вопросу, указывает на наличие у школьника необходимых теоретических знаний и практических навыков, окончательный ответ дан при правильном ходе решения, письменное решение аккуратно оформлено. Задание считается невыполненным, если ученик не приступил к его выполнению или допустил в нем погрешность, считающуюся в соответствии с целью работы ошибкой.

Оценивая результат обучения школьников по пятибалльной системе, учитель может пользоваться следующими примерными нормами оценок.

Положительная оценка («3», «4», «5») выставляется только в том случае,

когда ученик показал владение основным программным материалом. Оценка «5» выставляется при условии безупречного ответа либо при наличии 1—2 мелких погрешностей, «4» — при наличии 1—2 недочетов. Неудовлетворительная оценка («1» и «2») выставляется в том случае, когда ученик показал неусвоение основного программного материала.

Оценка за усвоение темы выставляется на основе всех текущих отметок. Особый вес придается оценкам за итоговую контрольную работу или ответы учащихся на зачетном занятии по всей теме. Однако итоговая оценка по теме должна отражать фактический уровень овладения материалом темы и не может выставляться как среднее арифметическое всех отметок. При выставлении тематической оценки учитель может не учитывать текущих отметок, если по результатам тематической контрольной работы или зачета эти отметки учащимися не подтверждены (например, неудовлетворительные оценки, полученные за пробелы в знаниях и умениях, которые затем были ликвидированы).

Годовая оценка должна отражать фактический уровень знаний учащихся на конец учебного года. При этом необходимо добиваться, чтобы учащийся заканчивал учебный год без существенных пробелов в учебной подготовке.

Важной задачей совершенствования контроля является привлечение учащихся к оценочной деятельности, организация на уроке их самоконтроля и самооценки. Необходимо, чтобы в ходе каждой проверки школьник не только узнал, чему он научился и чего не усвоил, какие допустил ошибки, но и осознал справедливость оценки, поставленной учителем, понимая, как можно самостоятельно оценивать свои ошибки. Для этого необходимо знакомить учащихся с критериями оценки, постепенно развивать умения содержательно оценивать свои знания и знания товарищей. В процессе обучения учитель обязан комментировать выставляемые оценки на основе критериев, сформулированных в программах, содержательно характеризовать требования к выполнению домашних заданий и классных самостоятельных работ, направляя учащихся к овладению различным учебным материалом на определенных уровнях познавательной самостоятельности, выделяя основные, наиболее важные вопросы. Такая четкая формулировка требований к знаниям и критериев их оценки воспитывает сознательное отношение школьников к учению, способствует осознанию и правильной оценке учащимися уровня своей учебной подготовки.

По мере оснащения школ кабинетами вычислительной техники, развития программного обеспечения школьных компьютеров все большую роль в качестве помощника учителя в его проверочно-оценочной деятельности будет играть ЭВМ. Наличие для нее специальных учебных программ (моделирующих, демонстрационных и тренажеров, автоматически фиксирующих результаты работ школьников), значительно облегчит текущую проверку состояния знаний класса, позволит проконтролировать умения и навыки всех учеников на каждом занятии.

Систематическая работа с ЭВМ на уроках информатики является важным фактором развития у школьников навыков самоконтроля. При отладке программ и других различных заданий компьютер автоматически фиксирует все ошибки учащихся; результаты его работы в точности соответствуют заданию и не всегда тому, что, может быть, имеет в виду ученик. Все это способствует развитию у школьников навыков аккуратного оформления программ и заданий, самостоятельному определению и исправлению допущенных ошибок и в конечном счете расширению и углублению их знаний.

Л. ЗВАВИЧ
Учитель-методист,
Л. САМОВОЛЬНОВА
Ведущий методист управления информатики и электронно-вычислительной техники Минпроса СССР

Телепередачи для девятиклассников

Телепередача «Этапы решения задач на ЭВМ». Это занятие отличается по форме и содержанию от предыдущих — здесь раскрываются возможности телевидения. Тридцать минут передачи посвящены экскурсии в вычислительный центр.

Далеко не все школы имели возможность организовать такую экскурсию, поэтому чрезвычайно важно дать учащимся возможность своими глазами увидеть современную вычислительную технику.

Учитель может заранее сообщить школьникам о форме передачи, предложить им быть как бы участниками осмотра, объявить конкурс на лучший конспект экскурсии. Возможно, стоит дать план этого конспекта:

примеры задач, решаемых на данном ВЦ;

какие упоминались науки;

в каких отделах ВЦ мы побывали;

что нам рассказали о машине БЭСМ-6;

что такое АЦПУ и какова история его развития;

как устроены графопостроители и цветная печать;

фамилии каких ученых, связанных с информатикой, были названы;

какие термины были непонятны.

Некоторые термины передачи окажутся незнакомыми многим зрителям, например гидротермодинамика, дифференциальные уравнения, локальные процессы и т. п. Учитель может впоследствии кратко разъяснить их или поручить отдельным ученикам заранее подготовить очень краткие сообщения на эти темы и выступить после передачи.

Подготовкой к телеуроку будет также изучение § 5 учебного пособия. Во всех отношениях полезно будет предложить учащимся выписать в тетрадь 5 этапов решения задачи с помощью ЭВМ; на них в начале передачи обращает внимание академик А. П. Ершов.

Школьники услышат имя Гурия Ивановича Марчука. На примере его научной, партийной и государственной деятельности раскрывается жизненный путь советского ученого. Г. И. Марчук занимается вычислительной и прикладной математикой, расчетом ядерных реакторов, динамикой атмосферы и океана, численными методами расчета автоматизированных систем управления. Он — физик и математик, президент АН СССР, делегат XXVII съезда КПСС, член ЦК КПСС, Герой Социалистического Труда, лауреат Ленинской и Государственной премий СССР.

Стоит обратить внимание школьников и на еще одно имя: зрители встречаются с оператором БЭСМ-6 Надей Поповой. Надя — вчерашняя школьница. Эту встречу можно использовать для профориентационной работы с учащимися.

Академик А. П. Ершов в этой передаче продолжает большой разговор о связи информатики с жизнью, наукой и различными школьными предметами. Он указывает на особую роль информатики в школе.

Телевизионная экскурсия даст возможность ученикам получить некоторое практическое представление о работе современного вычислительного центра.

Телепередача «Алгоритмы из курса математики» начинается с беседы в обстановке непринужденности и совместной интеллектуальной работы, которая должна царить на уроках информатики.

Авторы передачи колебались: готовить ли учащихся к съемке (оттачивать ответы, устранять ошибки и неточности, репетировать, переснимать) или снимать обсуждение таким, каким оно сложится.

Был избран второй путь. Конечно, у него есть и минусы. Многое зависит от того, как проходят уроки в вашем классе. Если свободное обсуждение задачи на уроке учитель не практикует, то будет нелегко убедить учащихся, что

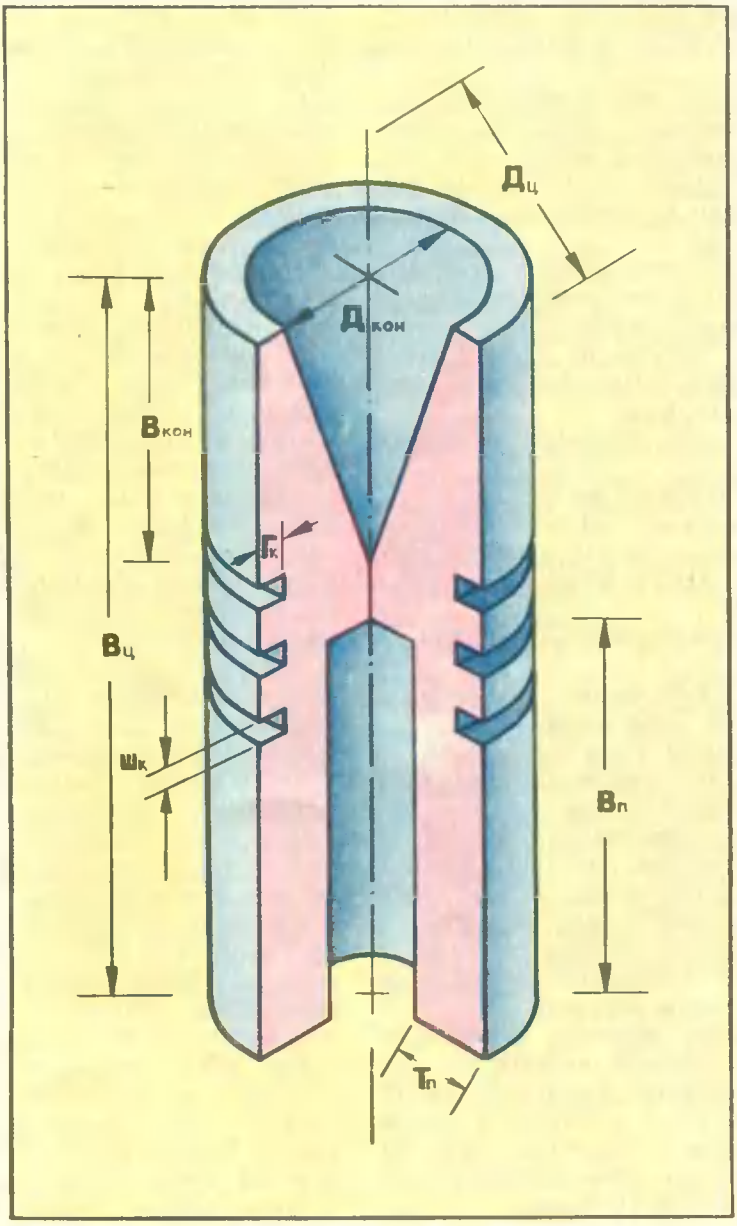
происходящее на экране не «разыграно как по нотам». Так или иначе, следует готовить учащихся к восприятию не только содержания телевизионной передачи, но и ее формы.

Основная цель телеурока — показать, как строятся алгоритмы математических задач. Будут рассмотрены задачи нахождения объема поршня и нахождения приближенного значения площади криволинейной трапеции.

Восприятие их решения усложняется тем, что они не встречаются еще в курсе математики IX класса; однако их анализ весьма полезен для уяснения ряда вопросов курса информатики.

Первая задача — нахождение объема поршня, представляющего собой цилиндр с конической и цилиндрической выемками и тремя канавками (рис. 1).

Еще до передачи надо напомнить уча-



щимся формулы объема цилиндра, конуса и цилиндрического кольца:

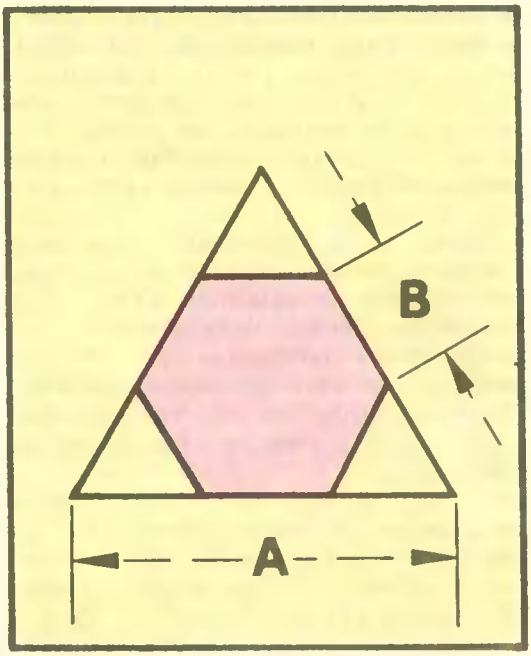
$$V_ц = \pi R^2 H = \frac{1}{4}\pi D^2 H$$

$$V_k = \frac{1}{3}\pi R^2 h = \frac{1}{12}\pi D^2 h$$

$$V_{ц\ кольца} = \pi R_1^2 h_{кольца} - \pi R_2^2 h_{кольца} = \pi (R_1^2 - R_2^2) h_{кольца}$$

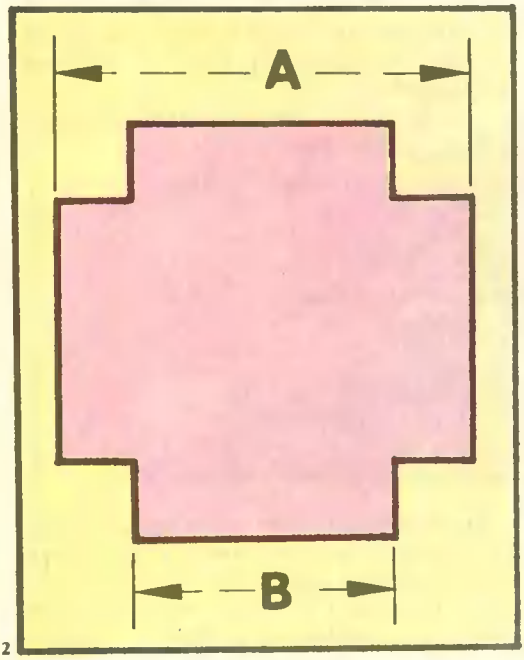
Заметим, однако, что для составления алгоритма нам вовсе не надо знать эти формулы: достаточно того, что их «знает» наша ЭВМ, т. е. в ней заложены алгоритмы ЦИЛИНДР, КОНУС, КАНАВКА. Наша задача — используя эти алгоритмы как вспомогательные, составить алгоритм ПОРШЕНЬ.

Возможно, будет полезно для понимания самой идеи решения рассмотреть несколько задач на нахождение площадей различных фигур. Например, для нахождения площади фигуры, изображенной на рис. 2, нужно из площади большого квадрата вычесть четыре площади малых квадратов; задача решается с помощью единственного вспомогательного алгоритма ПЛОЩАДЬ КВАДРАТА. Для нахождения площади фигуры, изображенной на рис. 3, нужен алгоритм ПЛОЩАДЬ ТРЕУГОЛЬНИКА, фигуры на рис. 4 — алгоритмы ПЛОЩАДЬ КВАДРАТА и ПЛОЩАДЬ ТРЕУГОЛЬНИКА.

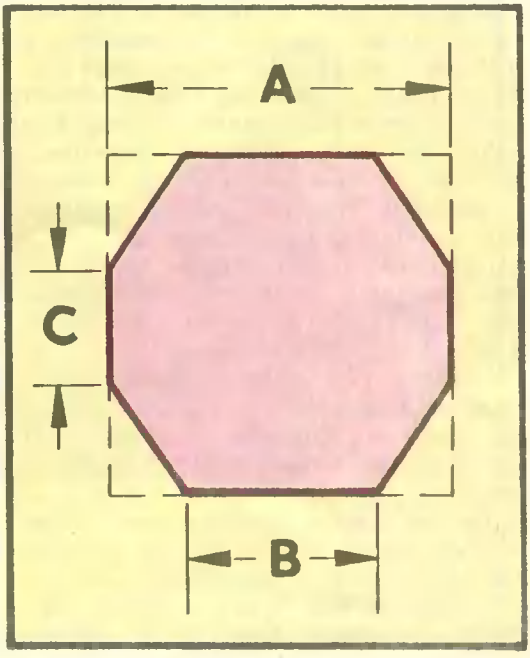


3

Поршень — фигура вращения. С сечениями таких фигур учащиеся в какой-то мере знакомы из курса черчения VIII класса. Возможно, стоит напомнить им, что осевое сечение цилиндра есть прямоугольник, конуса — равнобедренный



2



4

треугольник. Полезно показать на уроке осевое сечение цилиндрического поршня — это легко сделать с помощью кодоскопа. Желательно заранее начертить на доске изображение поршня. Если все эти подготовительные работы проведены, то соответствующая часть передачи будет легко усвоена.

Важно сконцентрировать внимание учащихся и на одном общем положении. В процессе решения задачи ведущий подчеркивает, что если алгоритм имеет много аргументов, то их надо располагать в осмысленном порядке. Формулируются два простых правила:

располагать рядом сходные величины, например:

$Д_ц$, $Д_{кон}$, $В_ц$, $В_{кон}$, $В_п$, $Т_п$, $Щ_к$, $Г_к$ (диаметр цилиндра, диаметр конуса, высота цилиндра, высота конуса, высота полости, толщина полости, ширина канавки, глубина канавки);

располагать рядом величины, относящиеся к одному и тому же вспомогательному алгоритму, например:

$Д_ц$, $В_ц$, $Д_{кон}$, $В_{кон}$, $Т_п$, $В_п$, $Щ_к$, $Г_к$.

В передаче школьники руководствуются первым правилом, но в качестве дополнительного варианта показано, как пользоваться вторым.

Вызовы вспомогательных алгоритмов отличаются от виденных в предыдущих передачах: среди их аргументов появятся не только числа и переменные, но и выражения. Так, для нахождения большего радиуса канавки надо диаметр поршня разделить пополам: для нахождения меньшего радиуса потребуется вычесть из большего глубину канавки.

Закончив решение задачи, ведущий еще раз подчеркивает, что при записи алгоритмов нужно заранее продумать обозначения и выписать все переменные, используемые в записи алгоритма, а не делать это постепенно.

Учитель сам решит, нужно ли ему заранее написать этот алгоритм на доске. Во время передачи школьникам будет довольно трудно воспроизвести запись в тетради; если же алгоритм будет у них все время перед глазами, например на экране кодоскопа, то восприятие улучшится, а перенести алгоритм в тетради можно будет и потом. Приведем алгоритм вычисления объема поршня, как он будет представлен на экране ($V_{цб}$ — объем большого цилиндра, $V_{цм}$ — объем малого цилиндра, $V_п$ — объем поршня):

```
алг ПОРШЕНЬ (вещ Дц, Дкон, Вц, Вп, Вкон,
Тп, Шк, Гк, Vп, Vцб, Vцм, Vкон, Vк)
    арг Дц, Дкон, Вц, Вп, Вкон, Тп, Шк, Гк
    рез Vп
нач вещ Vцб, Vцм, Vкон, Vк
    ЦИЛИНДР(Дц, Вц, Vцб)
    ЦИЛИНДР(Дц—2Тп, Вп, Vцм)
    КОНУС(Дкон, Вкон, Vкон)
    КАНАВКА(Дц/2, Дц/2—Гк, Щк, Vк)
    Vп:=Vцб—Vцм — Vкон—3Vк
кон
```

В передаче дается домашнее задание: записать алгоритмы вычисления объемов цилиндра, конуса, канавки;

взять какой-нибудь конкретный набор размеров поршня и исполнить алгоритм (как основной, так и вспомогательные).

Следующая задача — нахождение приближенного значения площади криволинейной трапеции — разобрана в учебнике информатики. Полезно рассмотреть с учащимися рис. 24 и 25 (с. 63 учебного пособия).

В алгоритме решения данной задачи используется конструкция команды «цикл». Нам нужно повторить цикл n раз, начав с X_0 и закончив X_{n-1}. В процессе составления этой конструкции часто возникают ошибки. На одну из них указано в передаче, как на наиболее типичную.

Алгоритм, рассматриваемый в телепередаче, имеет вид:

```
алг ПЛОЩАДЬ(вещ a, b,S, нат n)
    арг a, b, n
    рез S
нач вещ h, x, нат i
    S:=0; h:=(b—a)/n; i:=1; x:=a
    пока i≤n
        нц
        S:=S+f(x)·h
        x:=x+h
        i:=i+1
    кц
кон
```

Ведущий обращает внимание зрителей на команду присваивания первоначального значения промежуточной переменной i (счетчику команд) и на соответствующие особенности записи других

команд алгоритма. Если первоначально $i:=0$, то i — переменная типа цел, команды цикла будут повторяться, пока $i<n$. Если же $i:=1$, то i может быть переменной типа нат, а условие выполнения цикла будет записываться так: пока $i\leqslant n$.

Полезно исполнить алгоритм для примера $f(x)=x^2$, $x=0$; $x=1$, $n=9$ и сверить с приведенным результатом.

Итак, данная передача поможет нам увидеть, как в решении математических задач на ЭВМ реализуются вспомогательный алгоритм и конструкция «цикл»; она будет полезна учащимся не только для изучения информатики, но и как пропедевтический материал геометрии и математического анализа.

Телепередача «Алгоритмы из курса физики» проходит в виде лекции: строится алгоритм, демонстрирующий поведение тела на наклонной плоскости, и алгоритм расчета сопротивления электроцепи.

На уроке физики, если это возможно, или на уроке информатики перед просмотром передачи предстоит воспроизвести на доске рис. 5 и проделать расчеты для задачи о поведении тела на наклонной плоскости.

На тело, находящееся на наклонной плоскости, действуют сила тяжести

$|\vec{F}_1|=m_1g$,

натяжение нити

$|\vec{F}_{нт}|=m_2g$

и реакция опоры

$|\vec{F}_{оп}|=|\vec{F}_1|\cos\alpha$, вызывающая силу трения $F_{тр}$

От их соотношения будет зависеть состояние тела: покой, движение вверх или вниз. У нас возникла типичная ситуация алгоритма с командой ветвления, похожего на алгоритм исследования квадратного уравнения по дискриминанту; последний можно напомнить учащимся (с. 32 учебного пособия).

Что влечет тело вниз? Составляющая силы F_1, равная

$|\vec{F}_1|\sin\alpha=m_1g\sin\alpha$.

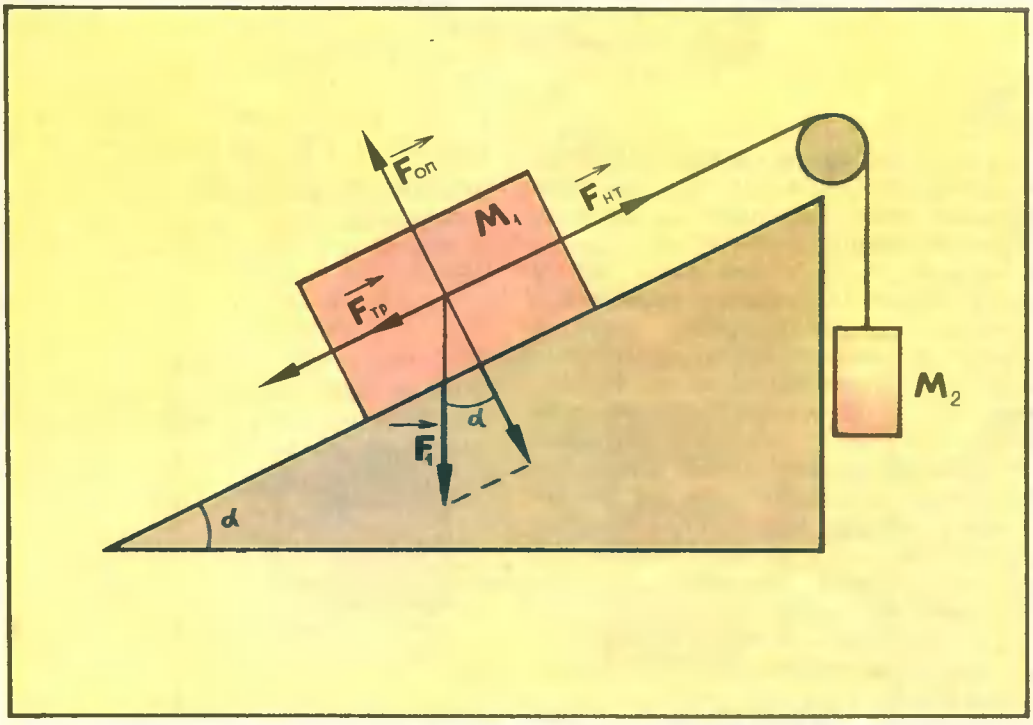
Что влечет тело вверх? Сила натяжения нити $|\vec{F}_{нт}|=m_2g$.

Если они равны, т. е.

$m_1g\sin\alpha-m_2g=0$,

то тело находится в покое.

Если они не равны, то возникает сила трения. В передаче для простоты предполагается, что коэффициент трения покоя и трения скольжения один и тот же.



5

Итак, $|\vec{F}_{оп}|=|\vec{F}_1|\cos\alpha$. Таким образом, $|\vec{F}_{тр}|=|\vec{F}_{оп}|\,k=k|\vec{F}_1|\cos\alpha$.

Если модуль разности $m_1g\sin\alpha-m_2g$ не превышает $\vec{F}_{тр}$, т. е.

$|m_1g\sin\alpha-m_2g|\leqslant k|\vec{F}_1|\cos\alpha$,

то тело находится в состоянии покоя (сила трения препятствует движению). Если $|m_1g\sin\alpha-m_2g|>k|\vec{F}_1|\cos\alpha$, то тело двигается, причем если $m_1g\sin\alpha-m_2g>k|\vec{F}_1|\cos\alpha$ (разность положительна), происходит движение вниз, а если $m_2g-m_1g\sin\alpha>k|\vec{F}_1|\cos\alpha$ (разность отрицательна) — вверх.

Приведем запись алгоритма, используемую в передаче (обозначение $f=|\vec{F}|$).

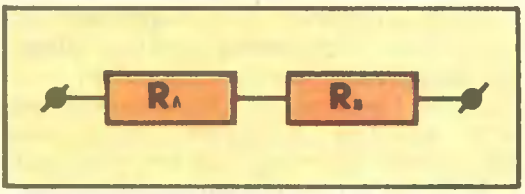
```
алг ПОВЕДЕНИЕ(вещ m1, m2, α, k, лит Y)
    арг m1, m2, α, k
    рез Y
нач вещ f1, f2, fтр, f
        f1:=9.8·m1
        f2:=9.8· m2
        fтр:=kf1· cos α
        МОД(f2—f1·sin α, f)
        если f<fтр
            то Y:=«покой»
              иначе
                    если f1·sin α>f2
                        то Y:=«вниз»
                        иначе   Y:=«вверх»
                    все
    все
кон
```

Рис. 5 и алгоритм следует поместить на доске, чтобы школьники во время передачи легче воспринимали происходящее на экране телевизора.

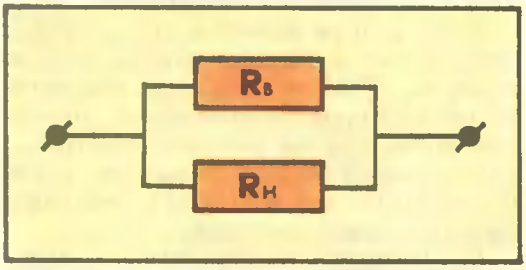
Академик А. П. Ершов в ходе объяснения задачи познакомит учащихся с термином «вычислительный эксперимент», так как исполнение этого алгоритма дает приблизительно те же знания, что и проведение эксперимента. Слушатели узнают, что на современной электронной машине можно продемонстрировать поведение тела не только в виде значений какой-либо литерной переменной, но и в виде «картинки», демонстрирующей движение тела по наклонной плоскости.

Для решения следующей задачи также требуется определенная подготовительная работа с учащимися.

Напомните девятиклассникам формулы для расчета сопротивления при последовательном и параллельном соединении проводников (рис. 6, 7).



6



7

Запишите с ними вспомогательные алгоритмы для последовательного соединения сопротивлений $R_л$ (левого) и $R_п$ (правого) — такие обозначения используются в передаче.

```
алг ПОС(вещ Rл, Rп, R)
    арг Rл, Rп
    рез R
нач
    R:=Rл+Rп
кон
```

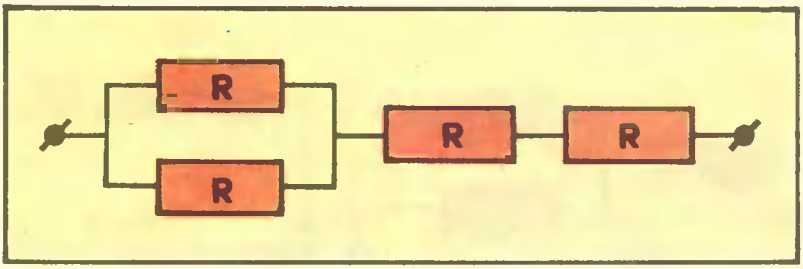
и для параллельного соединения $R_в$ (верхнего) и $R_н$ (нижнего)

```
алг ПАР(вещ Rв, Rн, R)
    арг Rв, Rн
    рез R
нач
    R:= Rв·Rн / (Rв+Rн)
кон
```

После этого можно рассмотреть порядок нахождения общего сопротивления схемы на рис. 8:

```
алг ЦЕПЬ(вещ r, R)
    арг r
    рез R
нач вещ x1, x2
    ПАР(r, r, x1)
    ПОС(x1, r, x2)
    ПОС(x2, r, R)
кон
```

Однако данную запись можно упро-



8

стить, оставляя без изменений число вызовов алгоритма, но меняя количество используемых переменных, например так:

```
алг ЦЕПЬ(вещ r, R)
    арг r
    рез R
нач вещ x
    ПАР(r, r, x)
    ПОС(x, r, x)
    ПОС(x, r, R)
кон
```

Не всегда, однако, можно сократить количество вспомогательных переменных до одной. Алгоритм вычисления сопротивления на схеме рис. 9 будет таким:

```
алг ЦЕПЬ(вещ r, R)
    арг r
    рез R
нач вещ x1, x2
    ПАР(r, r, x1)
    ПАР(r, x1, x2)
    ПОС(x1, x2, R)
кон
```

Аналогичная, но более сложная задача будет рассмотрена в передаче (рис. 10); эту схему стоило бы воспроизвести на доске перед передачей, но сам алгоритм в тетрадь не записывать.

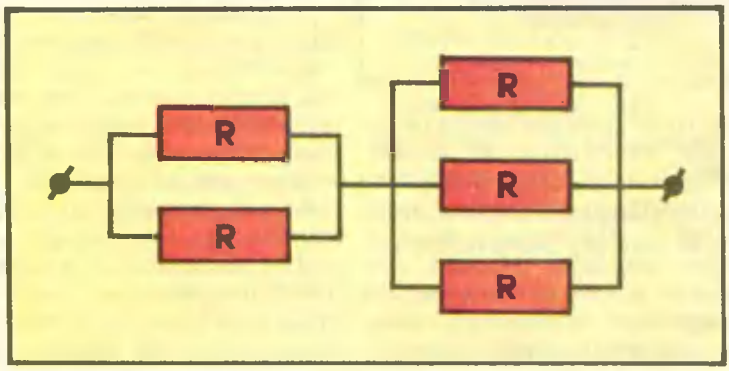
Думается, что на данном уроке максимальное внимание должно быть привлечено к сути читаемой лекции и демонстрируемым выкладкам. Если представится возможность, то результаты данного урока можно использовать на уроках физики.

После просмотра стоит еще раз напомнить ребятам о том, что говорилось в передаче, посвященной решению задач по математике. Если человек будет искать общее сопротивление цепи (рис. 11), он сделает это в одно действие: $R=R_1+R_2+R_3+R_4$.

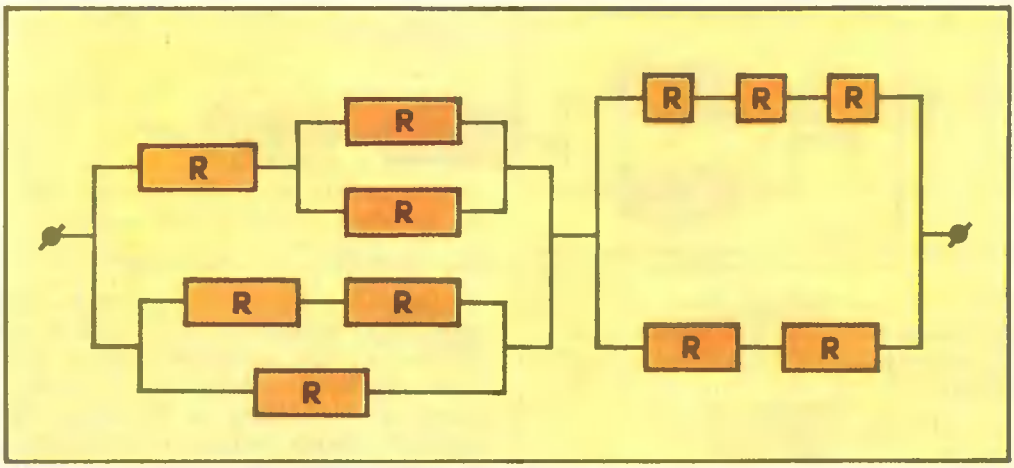
ЭВМ сделает это, три раза вызывая вспомогательный алгоритм:

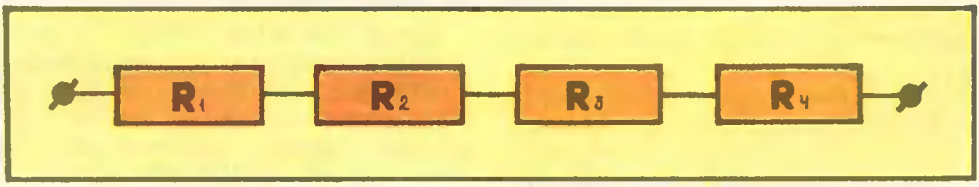
```
алг ЦЕПЬ(вещ R1, R2, R3, R4, R)
    арг R1, R2, R3, R4
    рез R
нач вещ x
    ПОС(R1, R2, x)
    ПОС(R3, x, x)
    ПОС(R4, x, R)
кон
```

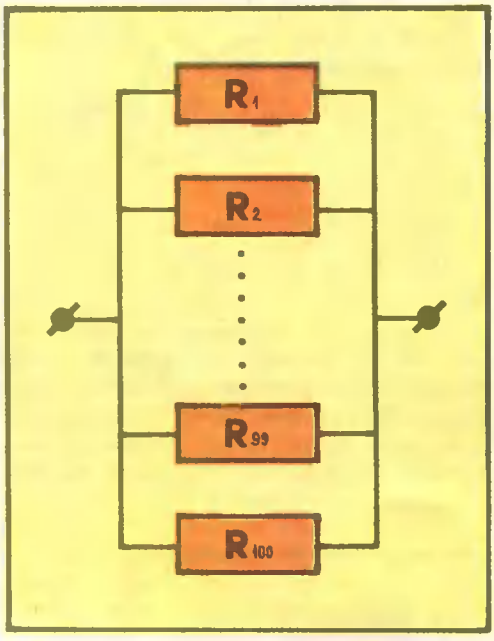
Можно при желании поставить вопрос об алгоритме нахождения общего сопротивления в цепи рис. 12, где соединены параллельно 100 сопротивлений. Алгоритм будет тогда содержать цикл. Можно попросить сильных учащихся составить его дома.



9







Итогом уроков, посвященных решению задач по математике и физике, может быть разговор о единообразии подходов к ним. Таким образом, предметная область информатики становится невероятно широкой. Можно сказать школьникам о том, что в принципе возможны передачи «Алгоритмы задач из курса химии», «Алгоритмы экономических задач», «Алгоритмы задач из курса биологии» и др. Полезно предложить ученикам, интересующимся тем или иным предметом, написать сценарий одной из таких передач (в том числе и по применению алгоритмов в математике и физике) и отослать его на телевидение.

Итак, мы рассмотрели три передачи по курсу информатики IX класса. Остается заметить, что просмотр всех передач дает довольно цельное представление об основных тенденциях курса. Ученик, внимательно просмотревший все передачи, заранее подготовленный к этому просмотру, расширяет свои знания по предмету.

По мере показа телеуроков в адрес авторов поступали замечания и предложения по совершенствованию передач. Они учитываются при работе над телециклом по курсу X класса. В будущем учебном году планируется дальнейшее совершенствование цикла учебных телепередач по всему курсу. Ждем замечаний и предложений по их организации, а также по методическим рекомендациям к их просмотру. Возможно, в какой-то школе нашли новый, более продуктивный способ использования этих передач. Авторы будут признательны за информацию об этом.

И. РОБЕРТ, Н. РАЗУМОВСКАЯ
НИИ ШОТСО АПН СССР

Практикум по курсу ОИВТ

Разработанная в НИИ ШОТСО АПН СССР программа предназначена в первую очередь для школ, оснащенных ПЭВМ «Агат» с языком программирования Бейсик-Агат, но может, с незначительными изменениями, использоваться и там, где КВТ оснащены другими ЭВМ.

Теоретическая часть курса изучается по пробному учебному пособию «Основы информатики и вычислительной техники». Практические занятия проходят 1 раз в неделю (по 1 ч); их цель — привить учащимся навыки общения с ПЭВМ на уровне пользователя- непрофессионала.

Желательно активнее использовать на занятиях графические средства программирования: они значительно облегчают восприятие учениками сложных конструкций (типа операторов цикла и условных операторов).

Для каждого занятия в программе определены тема, краткое содержание, приобретаемые знания, умения и навыки. Ориентировочно определено необходимое оборудование.

Занятия 1—2: понятие об аппаратном и программном обеспечении ЭВМ

Представление об основных устройствах ЭВМ, их назначении (центральный процессор, память; периферийные устройства: ввода, вывода, внешняя память). Программный принцип работы ЭВМ. Понятие об операционной системе, прикладных программах, трансляторах, интерпретаторах. Конструктивное исполнение ПЭВМ «Агат» (системный блок со встроенным дисководом, клавиатура, монитор, принтер). Ознакомление с рабочим местом учащегося, правилами техники безопасности. Последовательность включения ПЭВМ «Агат», начальная загрузка ДОС. Работа с демонстрационными, игровыми, учебными программами, загруженными и запущенными учителем.

Знать состав аппаратного и программного обеспечения ЭВМ; название и назначение основных устройств ПЭВМ «Агат». Уметь подготовить рабочее место к занятию: знать последовательность включения и выключения ПЭВМ; уметь производить начальную загрузку ДОС, знать назначение этой загрузки. Знать правила работы в кабинете и правила техники безопасности.

Оборудование: диски с демонстрационными, игровыми, учебными программами.

Занятие 3: алфавит версии языка программирования Бейсик; клавиатура ПЭВМ «Агат»

Буквы, цифры, знаки. Соответствие клавиатуры ПЭВМ алфавиту языка. Знакомство с клавиатурой: алфавитно-цифровые клавиши, управляющие, клавиши управления курсором.

Знать назначение, уметь пользоваться алфавитно-цифровыми клавишами, управляющими клавишами постоянного и временного переключения регистров (РУС, ЛАТ, РЕГ), клавишами «повторить» (ПВТ), «возврат каретки» (ВК). Знать назначение, уметь пользоваться клавишами управления курсором.

Оборудование: таблица «Клавиатура ПЭВМ «Агат», диски с программой «Клавиатура ПЭВМ «Агат».

Занятия 4—5: понятие о диалоговом режиме работы с ПЭВМ; классификация команд, операторов, функций

Принципы работы в диалоговом режиме. Режимы работы: командный, программный, режим ввода данных;

виды входной информации: команды; операторы; данные, используемые программой.

Реализация диалога в непосредственном режиме, программном режиме и режиме ввода: готовность ПЭВМ к вводу информации, ввод информации, реакция машины. Знакомство с системой команд ПЭВМ «Агат», классификация команд.

Знать особенности работы в диалоговом режиме. Знать виды входной информации и режимы работы с ПЭВМ. Знать назначение: знака], курсора, клавиши ВК, понимать смысл сообщения о синтаксической ошибке (SYNTAX ERROR). Уметь запустить загруженную программу (команда «запуск» — RUN), прервать работу программы (команда «сброс» — УПР—СБР). Уметь вводить данные, запрашиваемые программой; знать смысл сообщений об ошибках ввода (REENTER, EXTRA IGNORED).

Оборудование: диски с программами, предусматривающими ввод данных с клавиатуры.

Занятие 6: работа с гибким магнитным диском

Виды памяти — постоянная, оперативная, внешняя. Назначение различных видов памяти. Системные команды для работы с программами, находящимися на гибком диске: считывание и запись программ на диск. Команды: распечатка каталога, стирание программ с диска. Работа с демонстрационными и учебными программами, загруженными и запущенными учащимися.

Знать назначение постоянной и оперативной памяти; гибкого магнитного диска и дисковода; отличия внешней памяти от внутренней. Уметь вывести на экран каталог программ, хранящихся на диске (команда «каталог» — CATALOG). Уметь загрузить требуемую программу в память ПЭВМ (команда «загрузить» — LOAD), записать ее на диск под другим именем (команда «запись» — SAVE), удалить программу с диска (команда «уничтожить» — DELETE). Знать смысл сообщений об ошибках считывания (DISK I/O ERROR, FILE TYPE MISMATSH).

Оборудование: диски с программами (демонстрационными, игровыми, учебными).

Занятие 7: постоянные величины; вывод информации на экран; режим калькулятора

Числовые и текстовые константы. Вывод на экран, размещение на экране. Представление чисел (целые, вещественные в естественной и экспоненциальной формах), диапазоны значений. Арифметические операции, приоритет действий, скобок. Вывод результата.

Знать назначение и уметь использовать команду PRINT для вывода числовой и текстовой информации, результата вычислений. Уметь изменять цвет символов командой RIBBON. Знать запись и уметь использовать в режиме калькулятора целые и вещественные числа в естественной и экспоненциальной формах. Знать запись арифметических действий, приоритет действий и скобок; уметь записать и вычислить в режиме калькулятора арифметическое выражение. Знать смысл сообщений об ошибках «деление на 0» (DIVISION BY ZERO), «переполнение» (OVERFLOW ERROR).

Оборудование: таблица «Запись арифметических действий на языке программирования Бейсик».

Занятия 8—9: переменные; присваивание значений; функции; начальные сведения о программах

Числовые и текстовые переменные. Присваивание значений переменным. Стандартные математические функции. Работа в режиме калькулятора с использованием стандартных математических функций. Структура программ, нумерация строк, ввод программ.

Знать назначение, типы переменных, правила образования имен переменных. Уметь пользоваться командой присваи-

вания для задания числовых и текстовых значений переменным. Знать запись и уметь использовать в вычислениях стандартные математические функции. Знать смысл сообщения об ошибке «не тот тип величины» (TYPE ERROR). Знать определение программы, оператора, правила записи и ввода программной строки.

Оборудование: таблица «Примеры записи чисел в обычной форме и на языке программирования Бейсик», индивидуальные задания на вычисление выражений, содержащих стандартные математические функции.

Занятие 10: проверочная работа

Уметь подготовить ПЭВМ к работе, загрузить программу с диска и запустить ее. Уметь вводить данные, запрашиваемые программой; производить вычисления в режиме калькулятора с использованием стандартных математических функций. Понимать смысл изученных сообщений об ошибках.

Оборудование: индивидуальные задания для выполнения проверочной работы; диски с программами (учебными или демонстрационными, типа «Запуск спутника»).

Занятия 11—12: программный режим

Место программирования в решении задач. Элементы программ (операторы, данные, комментарии). Правила составления программ. Требования к программе. Составление и запуск простейших программ (вычисление значений выражений с использованием числовых и символьных переменных, стандартных математических функций; вывод текстовой информации на экран).

Знать этапы решения задач с помощью ЭВМ; основные элементы программ, правила записи комментариев (оператор REM). Знать назначение и уметь использовать в программе оператор «очистить экран» (HOME) и оператор окончания программы (END). Уметь организовать чтение из блока данных (операторы DATA—READ); знать правила составления и оформления программ, требования к программе.

Оборудование: таблица «Операторы языка программирования Бейсик».

Занятие 13: анализ работы программы; отладка программы

Ошибки записи. Логические ошибки. Диагностика ошибок. Вывод текста программы на экран. Распечатка программы. Редактирование программы. Удаление операторов (программных строк). Удаление программы из памяти.

Знать пути проверки правильности программы. Знать назначение и уметь пользоваться командами LIST (LIST, LIST N, LIST N, M), DEL, (DEL N, DEL M, N). Уметь редактировать текст программы путем замены программных строк или с помощью клавиш управления курсором. Знать назначение и уметь пользоваться командой «очистить память» (NEW). Уметь выводить программу (ее часть) на печать (команда LIST N/LIST N, M).

Оборудование: таблица «Операторы языка программирования Бейсик», диск с программой «D-100» (или «СРА», в зависимости от типа принтера).

Занятия 14—15: графические средства ПЭВМ «Агат»

Понятие о текстовой и графической информации. Режимы работы экрана ПЭВМ «Агат» — текстовой, графические (низкого, среднего и высокого разрешения). Графические примитивы: точка, линия, круг. Графические операторы. Построение геометрических фигур в различных графических режимах.

Знать виды графической информации, назначение и характеристики текстового и графических режимов. Уметь устанавливать требуемый режим (операторы GR=N, MGR=N, HGR=N, TEXT=N). Знать назначение и уметь пользоваться графическим оператором PLOT для задания точки, линии. Уметь строить геометрические фигуры в абсолютных координатах в различных графических режимах; уметь использовать оператор COLOR для задания цвета.

Оборудование: таблицы «Операторы языка программирования Бейсик», «Алгоритмы работы с графической информацией»*, диск с программами «Операторы Бейсика», «Графический редактор».

Задание 16: проверочная работа

Знать назначение изученных операторов и уметь использовать их при составлении простейших программ вычислительного характера, реализующих последовательные алгоритмы. Уметь составлять программы на построение геометрических фигур в абсолютных координатах в различных графических режимах. Уметь ввести и отладить на ПЭВМ «Агат» составленную программу.

Оборудование: индивидуальные задания для выполнения проверочной работы.

Занятие 17: циклы

Изменение порядка выполнения операторов. Организация бесконечного цикла. Генератор случайных чисел. Анализ простейшей демонстрационной программы, реализующей бесконечный цикл. Составление аналогичных программ.

Знать назначение и уметь использовать оператор GOTO для реализации бесконечного цикла. Уметь использовать генератор случайных чисел RND (1). Уметь, наблюдая работу простейшей демонстрационной программы, воспроизвести ее текст.

Оборудование: диски с демонстрационными программами (программа, рисующая лучи, исходящие из одной точки; «Лучи и спираль» и т. п.).

Занятия 18—20: циклы

Цикл с параметром: назначение, организация, примеры использования. Шаг цикла. Составление программы по предварительно подготовленным блок-схемам алгоритмов с использованием операторов цикла: демонстрационных в графическом режиме; вычислительного характера; на построение функций в заданном диапазоне значений.

Знать назначение оператора цикла. Уметь записать блок-схему и шаблон цикла на языке Бейсик—Агат. Знать назначение и уметь правильно задавать начальное и конечное значение параметра цикла, шаг цикла. Уметь использовать оператор цикла при составлении программ, реализующих циклические алгоритмы (организация движения точки на экране по вертикали, по горизонтали, по диагонали с помощью оператора PLOT в графическом режиме; поиск четных чисел от 2 до 50, вычисление суммы нечетных чисел от 1 до 50; построение графиков функций $y=ax+b$, $y=ax^2$ в пределах экрана и т. п.).

Оборудование: таблицы «Составные команды алгоритмического языка»*, «Способы записи алгоритма»*, «Блок-схема и шаблон цикла».

Занятия 21—22: организация диалога с программой

Язык диалога. Применение диалоговых программ. Техника программирования. Составление программ, реализующих диалог.

Уметь использовать операторы PRINT и INPUT для вывода сообщений и запросов на экран; операторы GET и INPUT для ввода числовой и символьной информации. Знать область применения и технику составления диалоговых программ (уметь организовать вывод на экран вопросов, строк-подсказок, запросов значений; ввод данных с клавиатуры).

Оборудование: диск с программой, реализующей вывод таблицы умножения и предусматривающей ввод множителя с клавиатуры; задания на составление диалоговых программ.

Занятия 23—24: организация диалога с программой

Изменение положения объекта на экране в графическом режиме. Динамика движения объектов. Изменение харак-

* Здесь и далее звездочкой отмечаются таблицы из учебно-наглядного пособия для IX класса (Кузнецов Э. И., Роберт И. Э. Алгоритмы и алгоритмический язык. Алгоритмы работы с величинами. М.: Просвещение, 1986).

тера движения в режиме диалога с программой.

Знать преимущества использования относительных координат, уметь описывать геометрические фигуры в относительных координатах, с помощью оператора PLOT. Уметь воспроизводить движение объекта в графическом режиме в системе координат экрана с использованием оператора цикла; организовать диалог с программой, реализующей движение объекта, предусматривающей запрос и ввод с клавиатуры шага перемещения.

Оборудование: таблица «Операторы языка программирования Бейсик».

Занятие 25: проверочная работа

Знать области применения диалоговых программ и уметь организовать диалог с программой. Уметь составлять программы на построение графиков функций, предусматривающие запрос и задание диапазона изменения аргумента в диалоговом режиме.

Оборудование: индивидуальные задания для проведения проверочной работы.

Занятия 26—29: ветвление

Операции отношения $<$, $>$, $<=$, $>=$, $<>$. Оператор условного перехода: назначение, состав, примеры использования. Составление разветвляющихся программ (по предварительно подготовленным блок-схемам алгоритмов): демонстрационных в графическом режиме; вычислительного характера; на построение графиков функций с диагностикой недопустимых значений аргумента и выхода функции за пределы экрана.

Знать назначение и уметь использовать оператор IF—THEN. Знать назначение операций отношения и уметь использовать их в операторе условного перехода. Уметь записать блок-схему ветвления и шаблон развилки на языке Бейсик—Агат с использованием операторов IF—THEN, GOTO. Уметь организовать выход из цикла по условию; использовать оператор условного перехода при составлении программ, реализующих разветвляющиеся алгоритмы (изменение направления движения объекта при определенном условии; решение квадратного уравнения; нахождение НОД; построение графиков функций типа $y=x^2-\frac{x}{2x+3}$ с выдачей сообщений при недопустимом значении x или при выходе функции за пределы экрана и т. п.).

Оборудование: таблицы «Составные команды алгоритмического языка»*, «Способы записи алгоритма*, «Блок-схема и шаблон ветвления».

Занятия 30—31: подпрограммы

Назначение, запись и расположение подпрограммы. Обращение к подпрограмме. Последовательность выполнения операторов. Составление программ, использующих обращение к подпрограмме, на основе предварительно подготовленных вспомогательных алгоритмов: демонстрационных в графическом режиме, вычислительного характера.

Знать преимущества использования подпрограмм, последовательность выполнения операторов при обращении к подпрограмме. Уметь записать подпрограмму и обратиться к ней из основной программы (оператор GOSUB), знать назначение оператора RETURH. Уметь использовать подпрограммы для реализации вспомогательных алгоритмов (воспроизведение движения объекта с использованием вспомогательного алгоритма описания контура объекта; нахождения НОК на основе вспомогательного алгоритма нахождения НОД; вычисление значения функции на основе вспомогательного алгоритма вычисления модуля числа и т. п.).

Оборудование: таблицы: «Библиотечные алгоритмы»*, диск с программой «Операторы Бейсика».

Занятие 32: повторение

Составление, ввод и отладка программ, реализующих последовательные, разветвляющиеся и циклические алгоритмы с использованием изученных операторов языка Бейсик—Агат.

Оборудование: таблицы «Операторы языка программирования Бейсик», «Составные команды алгоритмического языка»*, «Основные библиотечные алгоритмы»*.

Занятие 33: проверочная работа

Знать изученные команды и операторы, уметь пользоваться ими при составлении программ на языке Бейсик—Агат по предварительно подготовленным блок-схемам алгоритмов. Уметь задавать текстовый и графический режимы, знать их назначение и характеристики. Уметь воспроизводить текст программы, наблюдая ее работу (для простейших демонстрационных программ в графических режимах). Уметь осуществлять ввод и исполнение программ на ПЭВМ «Агат», распечатку текста и отладку программ.

Оборудование: индивидуальные задания для проведения проверочной работы.

Занятие 34: работа над ошибками; повторение пройденного материала

Оборудование: таблицы, шаблоны, диски с программами.

Программное обеспечение практикума

Программы-тренажеры

«Клавиатура «Агата» (Иглицкий А. М., НИИ ШОТСО АПН СССР, 1985).
«Операторы Бейсика» (Иглицкий А. М., НИИ ШОТСО АПН СССР, 1986).

Демонстрационные программы

«Графика «Агата» (Иглицкий А. М., с использованием фрагментов «Графического редактора» Яблонского А. К., НИИ ШОТСО АПН СССР, 1986).
«Графика «Агата» (256*256)» (Иглицкий А. М., НИИ ШОТСО АПН СССР, 1986).
«Графический редактор (256*256)»: (Березутский, 1986).
«Дорога» (Бадер Р., школа № 754 Москвы, 1986).
«Запуск спутника» (Разумовская Н. В., НИИ ШОТСО АПН СССР, 1985).
«Кубик Рубика», 1985.
«Лучи и спираль» (Роберт И. В., НИИ ШОТСО АПН СССР, 1986).
«Текстовый вывод» (Иглицкий А. М., НИИ ШОТСО АПН СССР, 1986).
«Теннис» (Голов А. А., 1986).
«Теннис» (Шевкунов В. В., НИИ ШОТСО АПН СССР, 1986).
«Шахматы» (Левин М. З., 1984).
«Шашки» (Левин М. З., 1984).

Математическое обеспечение

Дисковая операционная система 3.3. Интерпретатор языка Бейсик.

Р. АЛЕКСАНДРОВ
Черкасский педагогический институт

Основной алгоритм работы процессора

На курсах для преподавателей информатики в средних учебных заведениях выяснилось, что при овладении фактическим материалом II части пробного учебного пособия особое затруднение у учителей вызывает основной алгоритм работы процессора; видимо, и для учащихся он окажется сложным. Изучение же алгоритма важно потому, что оно призвано реализовать ведущую идею первого раздела программы по курсу информатики последнего года обучения — возможность автоматического выполнения программы; кроме того, «нестандартность» описания шагов алгоритма во фрагментах работы процессора удачно продолжает развитие алгоритмического мышления у учащихся.

Предложим ряд методических рекомендаций, облегчающих восприятие этого материала. Они прежде всего касают-

ся уточнений и детализации табл. 1—8 (с. 10—16 упомянутого пособия).

Табл. 2 служит для визуализации выполнения процессором программы решения задачи: числа 21 и 15 содержатся соответственно в регистрах R0 и R1; найти их сумму, поместив ее в регистр R2, и сохранить содержимое ячеек R0, R1. Имеет смысл табл. 2 оформить так:

Таблица 2

Шаг	Команда	СК (7)		Содержимое регистра		
		в начале шага	в конце шага	R0	R1	R2
		адрес любой	1500	21 (..0010101)	15 (..001111)	4 (..001000)
1	Переслать слово из R0 в R2	1500	1502	21	15	21
2	Добавить слово из R1 к R2	1502	1504	21	15	36
3	Стоп	1504	1506	21	15	36

Эта таблица согласована с предыдущей, содержит ответы на все вопросы задачи, напоминает о фактическом содержании регистров (двоичные коды) и содержании их на каждом шаге, отражает тот факт, что в начале и в конце работы процессора на каждом шаге содержимое СК разное. Понимание этих фактов особо существенно.

Следует четко определить те действия процессора, которые выполняются им только по командам перехода:

ЕСЛИ МЕНЬШЕ, ПЕРЕХОД НА +2 СЛОВА $\Rightarrow$ { при N=1 СК:=СК+2+4; при N=0 СК:=СК+2 }

Табл. 4 описывает программу для процессора к задаче «Большее из двух чисел R1 и R2 поместить в регистр R3». Заметим, что здесь отождествлены обозначения регистров и их содержимого. Если это нежелательно, то содержимое регистра RJ можно обозначить так: [RJ] (вспомним, что в геометрии часто отождествляют обозначения отрезка и его длины). Табл. 4 также следует согласовать с табл. 5 и 6 и дополнить ее сведениями о содержимом интересующих нас ячеек при выполнении условия R1<R2. В итоге табл. 4 примет такой вид.

Таблица 4

Шаг	Команды	Содержимое регистров при R1<R2			Содержимое регистров при R2≤R1		
		СК	R3	N	СК	R3	N
1	Сравнить слово R1 с R2	1500/1502	R3	1	1500/1502	R3	0
2	Если меньше, переход на +2 слова	1502/1508	R3	1	1502/1504	R3	0
3	Переслать слово R1 в R3		R3	1	1504/1506	R1	
4	Переход на +1 слово		R3	1	1506/1510	R1	
5	Переслать слово R2 в R3	1508/1510	R2			R1	
6	Стоп	1510/1512	R2		1510/1512	R1	

Корректировка табл. 5—8 осуществляется аналогично; в табл. 5 и 6 надо сохранить нумерацию шагов, принятую в табл. 4,— тогда пропуск команд станет наглядным и словесные пояснения в тексте окажутся лишними.

Е. ГРАНИЦА, Т. КРЮКОВА, Н. НЕСТЕРОВА, А. СЛУДНОВ
Канд. физ.-мат.- наук

И-86

Анализ изучения основ информатики и вычислительной техники в 1985/86 учебном году выявил острую необходимость создания практикума по этому курсу для эффективной подготовки и переподготовки преподавателей.

В ряде организаций АН СССР, АПН СССР, Минвуза СССР и Минпроса СССР, получивших первые КУВТ «Ямаха», были разработаны отдельные программы. Их приемка и отбор для использования в практикуме проводились созданной при Управлении ИВТ Минпроса СССР на базе ГИВЦ Минпроса СССР временной комиссией (председатель — И. М. Логвинова, отв. секретарь — Т. В. Крюкова). На основе отобранных комиссией программ сотрудниками НИИ ИВТ АПН СССР, МГУ им. М. В. Ломоносова, ГИВЦ и Управления ИВТ Минпроса СССР, НИИ ШОТСО АПН СССР создана первая версия набора программных средств для КУВТ «Ямаха» «Информатика-86» (И-86), обеспечивающего практикум на ЭВМ при переподготовке преподавателей курса ОИВТ (72-часовой спецкурс).

Набор включает:

программы по разделу «Устройство ЭВМ» (ЭВМ-практикум);

систему программирования на основе алгоритмического языка курса ОИВТ (Е-практикум);

тренажер для обучения алгоритмическому языку Рапира;

учебные программы по языку Бейсик;

обучающие программы по освоению клавиатуры, редактированию и ознакомлению с графическими возможностями ЭВМ;

демонстрационно-игровые программы.

Имеются также инструментальные программные средства:

база данных;

электронные таблицы;

операционная система (MSX DOS).

Всего в практикум включено 37 программ (см. табл. 1—3), располагающихся на трех дисках:

базовые программные средства по курсу ОИВТ;

демонстрационно-игровые программы;

инструментальные программные средства.

Все компоненты И-86 проверены на работоспособность в локальной сети. Проведен анализ и выработаны рекомендации по использованию программного обеспечения в конкретных разделах курса ОИВТ. Произведено тиражирование комплекта для всех союзных республик по числу имеющихся в них КУВТ «Ямаха». С представителями минпросов союзных республик на базе Ярославского ГПИ проведен двухдневный семинар, где они получили необходимые навыки работы с И-86 и рекомендации по подготовке и переподготовке преподавателей.

В НИИ ИиВТ АПН СССР проведены занятия с шестью группами преподавателей Новосибирской области (76 человек) с использованием И-86. Его эффективность подтверждается результатами анкетного опроса, показавшего значительное улучшение понимания преподавателями содержания предмета и методики его изучения.

Базовые программные средства для КУВТ «Ямаха» по курсу ОИВТ

Они включают:

Е-практикум (Е-86);

ЭВМ-практикум;

редактор текстов;

тренажер «Рапира»;

учебные программы по языку Бейсик;

обучающие программы для изучения графических возможностей Бейсика;

«азбука» — учебная программа по знакомству с клавиатурой;

MSX DOS — операционная система.

Е-практикум (разработка МГУ, Москва) — система программирования на основе алгоритмического языка курса ОИВТ, предназначена для ввода, редактирования и выполнения программ, а также задач и упражнений из пробного учебного пособия. Она имеет ряд ограничений: максимальная длина программы — 60 строк, максимальное число переменных и элементов массивов — 360; скорость исполнения программ около 30 операций в секунду. Созданные в Е-практикуме программы на диске не сохраняются; любая работа должна быть завершена в течение одного сеанса.

ЭВМ-практикум (разработка МГУ, Москва) — комплекс программ по разделу «Устройство ЭВМ» курса ОИВТ, имитирует работу микропроцессора. Практикум позволяет в непосредственном режиме вводить, редактировать и выполнять программы в кодах в таком же виде, как они описаны в пробном учебном пособии. Имитатор микропроцессора обрабатывает все 17 упомянутых там команд, работает с памятью объемом до 100 слов. Программы не сохраняются и должны быть завершены в течение одного сеанса работы в ЭВМ-практикуме.

Редактор текстов «лосьн» (разработка ИППИ АН СССР, Москва) позволяет освоить простейшие приемы редактирования текстов; возможно сохранение их на магнитных носителях. Предназначен главным образом для чтения и распечатки текстов, не являющихся программами для ЭВМ, хотя при выполнении определенных ограничений может использоваться и для ввода программ. Объем файла, с которым может работать редактор, около 600 знаков.

В файлах с инструкциями к редактору («описание 1» и «описание 2») вместо слов «редактор лось» следует читать «редактор лосьн».

Тренажер «Рапира» (разработка ВЦ АН СССР, Новосибирск) — эмулятор алгоритмического языка Рапира, в котором реализованы: структура данных; целочисленная арифметика над целочисленными объектами (числами, текстами, кортежами); простые предписания языка; простые управляющие и циклические структуры. Остальные возможности Рапиры не реализованы.

Программы не сохраняются и должны быть выполнены в течение одного сеанса.

Учебные программы по языку Бейсик (разработка ВЦ ЛГУ, Рига) предназначены для обучения основным элементам и конструкциям языка Бейсик.

Обучающие программы для изучения графических возможностей языка Бейсик (разработка НГПИ, г. Новосибирск), включающие в себя семь уроков по усвоению и закреплению знаний графических операторов системы MSX BASIC для КУВТ «Ямаха».

«Азбука» (разработка МИЭМ, Москва) — клавиатурный тренажер, предназначен для изучения и овладения клавиатурой КУВТ «Ямаха». Рекомендуется для первоначальных стадий обучения.

MSX DOS — операционная система, предназначена для работы с пакетом базовых программных средств, является стандартной, по сети не рассылается.

Пакет демонстрационных и игровых программ

Он предназначен для показа ряда возможностей ПЭВМ «Ямаха» на стадии начального освоения и содержит программы демонстрационно-игрового ха-

рактера, которые доступны для освоения всеми категориями пользователей, включает программы:

НГПИ1КЛ2.ИГР—НГПИ8КЛ2.ИГР;
ТАШИБА;
НАЧАЛО.MOS;
ИЛЛЮЗИЯ.MOS;
СЛОВА.MOS;
PREL2.MUS, FUGA3.MUS,
PALA1.MUS;
парк;
FP;
VERBS3.MOS;
NOM.

НГПИ1КЛ2.ИГР—НГПИ8КЛ2.ИГР — игровые контролирующие программы по арифметике и грамматике для младших школьников:

НГПИ1КЛ2.ИГР — восхождение на вершину пирамиды, предназначена для закрепления навыков умножения целых чисел;

НГПИ2КЛ2.ИГР — автомобильные гонки, предназначена для закрепления навыков вычитания целых чисел;

НГПИ3КЛ2.ИГР — парусная регата, предназначена для закрепления навыков вычитания целых чисел;

Таблица 1

Базовые программные средства по курсу ОИВТ (диск № 1)

Программа	Файл		Номера поддерживаемых программой разделов		Время загрузки по сети	Загрузка и запуск программы	Остановка программы
	имя на диске	содержимое	курса ОИВТ	спецкурса			
Е-практикум (Е-86)	E86NET.BAS	программы	1	2	30 с	см. табл. 4	RESET
	E86.COM	загрузки					
	E86.DOK	инструкция					
ЭВМ-практикум	PDPBOOT	программы	12	3	40 с	см. табл. 5	RESET
	PDPSENT	запуска					
	PDPNET1	работы					
	PDPNET2	ЭВМ-практикума					
	PDP11TTL	инструкция					
	PDP11.DOC	инструкция					
Простой редактор текстов	лось н	редактор	1, 4	2, 7	около 2 мин	см. табл. 6	RESET
	описание. 1	инструкция					
	описание.2	инструкция					
Тренажер «Рапира»	MICRO.COM	эмулятор	6	6	около 4 мин	см. табл. 7	RESET
	BLOAD, BSAVE	программы					
	MICRO1.COM	запуска,					
	MICRO2.COM	работы тренажера					
	RESEIV.OBJ	инструкция					
	рапира.TXT	инструкция					
	микро.TXT	инструкция					
Практикум по Бейсику	BASIC1	программа	6	6	2 мин 30 с	см. табл. 6	CTRL+ +STOP
	BASIC2	программа					
Практикум по Бейсику (графические возможности)	BASGR	программы	1	6	30— 50 с	см. табл. 6	CTRL+ +STOP
	BASGR.L01	для изучения					
	BASGR.L02	графических					
	BASGR.L03	возможностей					
	BASGR.L04	Бейсика					
	BASGR.L05						
	BASGR.L06						
	BASGR.L07						
Клавиатурный тренажер	азбука	программа	3	3	около 2 мин	см. табл. 6	см. инструктаж на на экране ПЭВМ
	азбука.рук	инструкция					

Таблица 2

Демонстрационно-игровые программы (диск № 2)

Программа	Файл		Номера поддерживаемых программой разделов		Время загрузки по сети	Загрузка и запуск программы	Остановка программы
	имя на диске	содержимое	курса ОИВТ	спецкурса			
По арифметике и грамматике	НГПИ1КЛ2.ИГР НГПИ2КЛ2.ИГР НГПИ3КЛ2.ИГР НГПИ4КЛ2.ИГР НГПИ5КЛ2.ИГР НГПИ6КЛ2.ИГР	игровые программы по арифметике (для начальных классов)	17	8	15—20 с	см. табл. 6	CTRL± +STOP
	НГПИ7КЛ2.ИГР НГПИ8КЛ2.ИГР	программы по грамматике					
Графическая	ТАШИБА	программа демонстрации графических возможностей ЭВМ «Ямаха»	17	8	1 мин 40 с	см. табл. 6	CTRL+ +STOP
Работа с ЭВМ	НАЧАЛО.MOS	программа	9, 17	8	40 с	см. табл. 6	CTPL+ +STOP
Демонстрационно-игровая для сравнения длин отрезков	ИЛЛЮЗИЯ.MOS	программа	9	8	15 с	см. табл. 6	ответ на запрос ПЭВМ «Н» (нет)
Клавиатурный тренажер	СЛОВА.MOS	программа	3, 9	3	20 с	см. табл. 6	CTRL+ +STOP
Музыкальные	PREL.MUS	программа	9, 17	8	25 с	см. табл. 6	CTRL+ +STOP
	FUGA3.MUS PALA1.MUS	программа программа			20 с 15 с		
Игровая	парк	программа	9, 17	8	в сеть не загружается	bload «парк», r	RESET
Учебная по тригонометрии (контроль знаний формул приведения)	FP	программа	13	8	30 с	см. табл. 6	самоотключение
Учебная по английскому языку (правила спряжения)	VERBS3.MOS	программа	17	8	30 с	см. табл. 6	самоотключение
Учебная по физике (контроль знания закона Ома)	NOM	программа	13	8	1 мин 40 с	см. табл. 6	

НГПИ4КЛ2.ИГР — окраска прямоугольника, предназначена для закрепления навыков сложения целых чисел;

НГПИ5КЛ2.ИГР — стыковка космических кораблей, предназначена для закрепления навыков деления целых чисел;

НГПИ6КЛ2.ИГР — погоня волка за зайцем, предназначена для закрепления навыков деления целых чисел;

НГПИ7КЛ2.ИГР — прыжок дельфина через кольцо, предназначена для закрепления знаний по грамматике;

НГПИ8КЛ2.ИГР — погоня акулы за рыбкой, предназначена для закрепления знаний по грамматике.

Программы имеют 9 уровней скорости ответа (разработаны НГПИ, Новосибирск).

ТАШИБА — демонстрационная про-

грамма для показа графических возможностей ПЭВМ «Ямаха».

НАЧАЛО.MOS — обучающая программа, позволяющая сделать первые шаги в общении с ЭВМ (разработана в ТГУ, Тарту).

ИЛЛЮЗИЯ.MOS — игровая программа о сравнении длин отрезков (разработка в ТГУ, Тарту).

СЛОВА.MOS — демонстрационно-игровая программа для отработки навыков работы с клавиатурой при пер-

Таблица 3

Инструментальные программные средства (диск № 3)

Программа	Файл		Номера поддерживаемых программой разделов		Время загрузки по сети	Загрузка и запуск программ	Остановка программы
	имя на диске	содержимое	курса ОИВТ	спецкурса			
Программный комплекс МИКРОМИР-86 Электронные таблицы	MIM*.*	программы	1	7		см. табл. 8—9	
	MIMDOC.DOC	инструкция					
	mihmat.doc	инструкция					
	MP.COM	программы	8	7	в сеть не загружается	в системе MSX DOS A>MP	Q<
	MP.80	программы					
	MP.COD	инструкция					
	MP.HLP	инструкция					

Примечание. Приводится описание только МИКРОМИРа-86 и электронных таблиц, так как остальные файлы предназначены для системного программиста. В МИКРОМИР-86 входят все файлы дискеты, начинающиеся на символы MIM. Доступ к инструкциям MP.COD, MP.HLP возможен только во время нахождения в системе MVLTIPLAN (нет вывода на PRINT).

Таблица 4

Загрузка Е-практикума по локальной сети

Действие	Нажмите клавиши на учительской ПЭВМ	Изображение на экране дисплея учительской ПЭВМ после выполнения команды	Нажмите клавиши на ученической ПЭВМ	Изображение на экране дисплея ученической ПЭВМ после выполнения команды
Включить сеть			RESET	run «COM:»
Загрузить MSXDOS		A>		
Перейти в Бейсик	basic	Ok		
Загрузить программы пересылки	load «E86NET.bas»	Ok		
	run 100	Ok		
	run110	A>		
Переслать Е86 на ученические ПЭВМ	E86	заставка Е-практикума на учительской ПЭВМ		
	SHIFT+ CTRL+ +SELECT			Заставка Е-практикума на ученической ПЭВМ (через 30 с)

Примечание. В конце каждого текста, набираемого на учительской ПЭВМ, следует нажать клавишу ввода («изогнутая стрелка»).

Таблица 5

Загрузка ЭВМ-практикума по локальной сети

Действие	Нажмите клавиши на учительской ПЭВМ	Изображение на экране дисплея учительской ПЭВМ после выполнения команды	Нажмите клавиши на ученической ПЭВМ	Изображение на экране дисплея ученической ПЭВМ после выполнения команды
Включить сеть			RESET	run «COM:»
Загрузить MSX DOS		A>		
Перейти в Бейсик	basic	Ok		
Переслать на ученические ПЭВМ	load «PDPBOOT»	Ok		
	save «COM:»	Ok		
	run «PDPSEND»	Ok		Рамка ЭВМ-практикума (через 40 с)

Таблица 6

Загрузка по локальной сети стандартным способом в системе MSX BASIC

Действие	Нажмите клавиши на учительской ПЭВМ	Изображение на экране дисплея учительской ПЭВМ после выполнения команды	Нажмите клавиши на ученической ПЭВМ	Изображение на экране дисплея ученической ПЭВМ после выполнения команды
Включить сеть				
Загрузить MSX DOS		A>		
Перейти в Бейсик	basic	Ok		
Загрузить программу (например, с именем «имя»)	load «имя»	Ok		
Передать программу по сети	save «COM:»	Ok		Заставка программы

воначальном знакомстве с ЭВМ (разработана в ТГУ, Тарту).

PREL2.MUS, FUGA3.MUS, PALA1.MUS — демонстрационно-музыкальные программы (разработаны в ТГУ, Тарту).

Парк — игровая программа фирменного обеспечения с музыкальным сопровождением. Включена по рекомендации ГИВЦ МП СССР.

FP — программа-тренажер для усвоения формул приведения на уроках по тригонометрии (разработана в НИИ ШОТСО АПН СССР).

VERBS3.MOS — обучающая программа по английскому языку, предназначена для закрепления навыков по спряжению глаголов в Present Indefinite (разработана в ТГУ, Тарту).

NOM — учебная программа, предназначенная для закрепления знаний при изучении закона Ома на уроках физики. Имеет три уровня сложности (разработана в НИИ ШОТСО АПН СССР).

Пакет инструментальных программных продуктов

Пакет включает в себя следующее стандартное фирменное инструменталь-

Таблица 7

Загрузка тренажера с языка Рапира по локальной сети

Действие	Нажмите клавиши на учительской ПЭВМ	Изображение на экране дисплея учительской ПЭВМ после выполнения команды	Нажмите клавиши на ученической ПЭВМ	Изображение на экране дисплея ученической ПЭВМ после выполнения команды
Включить сеть			RESET	run «COM:»
Загрузить MSX DOS		A>		
Перейти в Бейсик	basic	Ok		
Загрузка программы пересылки	load «BLOAD» save «COM:» run «BSAVE»	Ok Ok A>		
Загрузка программы тренажера	MICRO Д	ПЕРЕДАВАТЬ ПО СЕТИ? ПЕРЕДАЮ: ЖДИТЕ ТРИ МИНУТЫ Заставка		Заставка

Таблица 8

Загрузка МИКРОМИРа-86 по локальной сети

Действие	Нажмите клавиши на учительской ПЭВМ	Изображение на экране дисплея учительской ПЭВМ после выполнения команды	Нажмите клавиши на ученической ПЭВМ	Изображение на экране дисплея ученической ПЭВМ после выполнения команды
Включить сеть			RESET	run «COM:»
Загрузить		A>		
Перейти в Бейсик	basic	Ok		
Пересылка МИКРОМИРа в сеть	load "MIMNETS". BAS" run 100 run 110	Ok (через 15 с) Ok файл >		
Переслать редактор вместе с рассылаемым файлом (например, MIM.HLP) на ученические ЭВМ	MIM.HLP ESC →	Текст файла		Изображение текста файла (через 30 с)
Для временного выхода в MSX DOS (если необходимо)	SHIFT+F4	A>		

ное математическое обеспечение:

система программирования на основе алгоритмического языка С;

Турбопаскаль (быстрый Паскаль) — система программирования на основе алгоритмического языка Паскаль:

системы программирования на основе ассемблеров;

SCED — система редактирования;
MSX DOS — операционная система;
DATABASE — база данных;

Таблица 9

Выгрузка из сети ученических файлов в МИКРОМИРе-86

Действие	Нажмите клавиши на учительской ПЭВМ	Изображение на экране дисплея учительской ПЭВМ	Нажмите клавиши на ученической ПЭВМ	Изображение на экране дисплея ученической ПЭВМ после выполнения команды
Войти в файл Подготовиться к приему	ESC←	Появление текста файла (через 30 с)	ESC↑	
Сохранение файла Для временного выхода в MSX DOS	ESC↑ SHIFT+F4	A>		

MULTIPLAN — электронные таблицы.

Кроме того, в пакет входят нестандартная (с возможностью загрузки по сети) операционная система MSX DOS, разработанная в УПЦВТ Октябрьского р-на г. Москвы, и интегрированная программная среда МИКРОМИР-86, разработанная в МГУ (Москва).

МИКРОМИР-86 — интегрированный программный комплекс. Его программная среда включает в себя текстовый редактор, текстовый процессор и редактор каталога файлов с простейшими инструментальными возможностями.

Пользуясь МИКРОМИРом-86, можно освоить простейшие приемы редактирования текста на ЭВМ, а также более сложные операции по обработке текстов (поиск, форматирование и др.) и методику организации личного архива.

В процессе обучения рекомендуется использовать редактор МИКРОМИРа-86 в одном из двух режимов: без сохранения и с сохранением результатов.

В первом режиме учитель может разослать всем ученикам редактор вместе с одним и тем же редактируемым текстом (файлом). Ученики осваивают редактор, но результаты редактирования не сохраняются на диске.

Во втором режиме учитель посылает по сети индивидуально каждому ученику редактор с его персональным текстом (файлом). По окончании редактирования учитель сохраняет результаты работы каждого ученика на диске. Процедура загрузки ученической машины занимает не более минуты.

В процессе работы МИКРОМИРа-86 с сетью дискета не должна меняться, так как при смене оглавление новой дискеты будет испорчено и содержимое ее станет недоступным.

По сети можно переслать файл объемом не более 32 000 символов, при попытке переслать файл большего объема программа выдает отказ. Если в процессе работы ученика произойдет выход за пределы объема файла, то редактирование завершится аварийно и результаты работы пропадут.

Полная инструкция приводится на дискете в файловых документах. МИКРОМИР-86 обладает наиболее полным редактором текстов отечественного производства с разнообразными возможностями работы. Он доступен пользователям различного возрастного и образовательного уровня.

MULTIPLAN (электронные таблицы) — система представления и обработки данных, дающая возможности для представления, интерпретации и преобразования информации. Исходная единица информации рассматривается как ячейка таблицы. Ссылка и идентификация ячейки осуществляется по имени или по номерам ряда и колонки. Ячейки можно группировать в виде ряда, колонки или группы. Описав взаимосвязь между ячейками в виде формул и представив информацию как таблицу, пользователь получает возможность ме-

нять отдельные значения и прослеживать цепочку изменений в результатах. Все это демонстрируется в таблице, представленной на экране. Он вырезает часть поля ячеек, двигаясь в соответствии с командами пользователя по всей таблице. Наиболее интересные участки таблицы можно поместить в неподвижное окно на экране. В другой части окна, перемещаясь по всему полю таблицы, можно производить нужные пользователю корректировки.

Система электронных таблиц имеет встроенный набор команд работы с ячейками (ссылки, форматы, копирование, переходы к группе ячеек, присвоение символических имен), производит обслуживание файлов (ввод с диска, вывод таблицы на диск), вычисляет различные функции (синус, косинус, сумма, среднее значение, вариация значений ячеек от одного пересчета к другому). Информационные ячейки могут содержать числовую или текстовую информацию, что повышает наглядность таблиц при использовании текстовых ячеек как комментариев к числовым.

Система работает под управлением операционной системы MSX DOS с настройкой на определенный терминал.

Система выдает справки о командах, вызываемых во время обращения к текущей команде. Вызов справочной информации осуществляется нажатием клавиши «?».

Широкое тиражирование и использование комплекта И-86 дает возможность организовать подготовку и переподготовку преподавателей курса ОИВТ на более высоком уровне. Данный практикум позволит улучшить понимание учителями содержания и предмета курса ОИВТ и методики его преподавания.

Кроме того, И-86 может быть использован как практикум по курсу ОИВТ для учащихся IX—X классов, так как содержит в себе необходимые программные средства для усвоения основных положений курса и приобретения важнейших навыков работы с ЭВМ.

В связи с наличием в И-86 демонстрационных, игровых и обучающих программ, иллюстрирующих возможность использования ЭВМ при изучении общеобразовательных предметов, он может быть использован в качестве основы для создания практикума по мини-всеобучу учителей-предметников.

Очевидна необходимость проведения работ по дальнейшему развитию И-86. Очередная его версия будет распространена к учебному году.

Замечания и пожелания по составу И-86 и отдельным программам просьба направлять в Управление ИВТ Минпроса СССР по адресу: 119034, Москва, Смоленский бул., 4.

А. АРХАНГЕЛЬСКИЙ

Мир ЭВМ

Есть особое очарование, которое испытывает каждый работающий на небольшой отдельной ЭВМ. Оно рождается ощущением полной власти над целой вычислительной системой и сознанием того, что все можно попробовать, все можно сделать. Пионеры ЭВМ, создавая и используя первые машины, почувствовали это очарование. Оно передалось и современным программистам.

Некоторые специалисты могут возразить, что такое ощущение сродни ностальгии и неуместно в замкнутом мире машинных операций. Они утверждают, что средний пользователь совершенно удовлетворен программированием на языке высокого уровня и счастлив, что ему удается игнорировать большинство аспектов оборудования, программного обеспечения и системы в целом.

Эта статья — не для них. Она для тех, кто, будучи очарован машиной, желает понять «магию», окружаю-

щую применяемые устройства. Такие пользователи (программисты, ученые, инженеры, специалисты по прикладным вопросам) хотят знать несколько больше об увлекательной жизни системы. Эта статья для тех, кто не хочет иметь поражений, несмотря на неполадки аппаратуры, сбои программ операционной системы и подчас полное отсутствие документации.

Все формы взаимодействия человека с ЭВМ можно разделить на несколько уровней.

Один пользователь без файловой системы — это, например, работа с калькулятором или персональным компьютером без внешних запоминающих устройств.

Один пользователь с файловой системой — это, например, работа на микро- или мини-ЭВМ с развитой внешней памятью на дисках или магнитных лентах. Неважно, выполняет ли машина одну или одновременно несколько задач. Главное — появляются понятия файла как единицы информации и, соответственно, операции над файлами.

Много пользователей с одной файловой системой — это, например, работа на мини-ЭВМ с развитыми внешней памятью и устройствами ввода-вывода, позволяющими работать одновременно нескольким пользователям. В дополнение к предыдущему уровню появляются понятия о защите информации одного пользователя от преднамеренных или непреднамеренных действий другого, об администраторе системы. Так как процессор в конкретный момент времени может выполнять только одну задачу, а в системе выполняется много задач многих пользователей, более остро встает вопрос о дисциплине распределения времени процессора, оперативной памяти и памяти на внешних носителях между задачами и пользователями.

Много пользователей, много файловых систем — такая ситуация может возникнуть, например, при работе в сети нескольких ЭВМ. При этом особенно повышаеся ответственность пользователя за свое поведение в общей системе.

Здесь будут рассматриваться системы на базе ЭВМ ДВК-2М и Электроника-100/25 и операционной системы RT-11. Несмотря на конкретизацию используемой ЭВМ и операционной системы, наша цель — умение анализировать возникающие ситуации, мыслить категориями системы, а оно не зависит ни от типа операционной системы, ни от типа ЭВМ.

Мы преднамеренно говорим не о вычислительной системе, а о системе вообще; ведь современные ЭВМ занимаются не только вычислениями. Другая причина в том, что в статье под системой понимается не только и не столько организация аппаратной части и операционной системы, сколько организация логических связей и взаимодействие программ с аппаратурой и между собой, а также роль и поведение человека в этой среде.

1. Давайте познакомимся

1.1. Мир, где живет ЭВМ, или Физический мир системы

Мир — огромный, прекрасный, где живут люди, называется планетой Земля. Все, что нас окружает, все, с кем мы общаемся,— это наш мир.

Быстротечный мир фантазии и мечты, льющийся к нам с экрана,— это другой, но тоже реальный мир, мир кинематографа.

Для тех читателей, которые уже знакомы с вычислительной техникой и стремятся углубить свои знания в этой области, мы начинаем публикацию руководства по использованию распространенной на персональных ЭВМ операционной системы. Поясним некоторые термины.

Слово *система* используется здесь во многих смыслах. Комплекс электронных устройств, составляющих ЭВМ, и программное обеспечение этого комплекта составляют систему; ее обычно называют вычислительной. Программное обеспечение имеет некоторое ядро, управляющее остальными программами,— его называют операционной системой. Системой является и «голая», незапрограммированная ЭВМ — ведь она состоит из множества взаимодействующих устройств! И вот еще одна система: фай-

И ЭВМ должна где-то «жить», с чем-то общаться. Мир ЭВМ ограничен оборудованием, которое с ней тем или иным способом связано.

В первую очередь подключаются к ЭВМ устройства ввода-вывода информации. Это дисплей, на котором высвечиваются символы, графический дисплей, на экране которого можно рисовать рисунки. Есть и настоящий чертежный стол, где рисует сама ЭВМ — он называется графопостроителем. Ну и, конечно, клавиатура.

Следующая группа — различные устройства внешней памяти, большие и маленькие, быстрые и медленные.

Подобных устройств придумать можно много, и придумывают их постоянно. ЭВМ окружает такое множество вспомогательных устройств, что саму ее уже не видно. Все это вместе называется аппаратной частью (hardware) и составляет физический мир системы.

1.2. Что окружает ЭВМ, или Конкретная реализация системы

Первые машины PDP-11 корпорации DEC появились в начале 70-х гг. В отличие от серии PDP-8, представляющей собой многократное повторение по существу одной и той же машины в разных технологических вариантах, семейство PDP-11 задумано и осуществляется как ряд машин, объединенных единой архитектурой, но значительно отличающихся друг от друга в отношении производительности и цены. Семейство PDP-11 неуклонно расширяется и включает уже более 20 моделей, покрывающих обширный диапазон мощностей — от микрокомпьютера LSI-11, поставляемого в минимальном варианте с 8K байт памяти, до «супермикро» MIKRO-PDP-11/83 с памятью до 4M байта, производительностью на уровне средних моделей IBM S/360—370.

Замечательной чертой этого семейства является и тщательно соблюденная на всех уровнях совместимость аппаратуры и программного обеспечения, а также кодов и форматов представления данных на различных носителях.

Во многих странах выпускаются как аналоги, так и машины, архитектурно совместимые с PDP-11. Это и румынские CORAL 4001/4011/4030, и венгерские JANUS (последние большей частью используют советские микропроцессоры и другие комплектующие изделия). Среди советских машин получила всеобщее признание серия малых ЭВМ (СМ ЭВМ), включающая СМ-3, СМ-4, СМ-1420, а также Электронику-60, Электронику-100/25, Электронику-79, МС1211, МС1212, НМС1201, ДВК-2М. Мы подробно изучим Электронику-100/25 и ДВК-2М.

Итак, соберем комплект аппаратуры для системы с минимальными возможностями, который выпускается нашей промышленностью под названием ДВК-2М.

Процессор. Для выполнения программ нам необходимы арифметико-логическое устройство, оперативная память и устройство управления — здесь это будет одноплатная микро-ЭВМ Электроника НМС1201.

Терминал. Для общения с системой нам необходим терминал, который состоит из клавиатуры и, в нашем случае, из дисплея. Мы будем использовать 15ИЕ-00-013, но можно использовать и дисплей венгерского производства Видеотон VDT52130.

Внешняя память. В качестве внешней памяти в малых системах используются накопители на гибких

ловая. Так называют особым образом организованный массив информации. Если сравнить ее с библиотекой, то ее составляющие — файлы — это книги. Файлы состоят из записей, которые похожи на фразы, хотя зачастую и содержат только цифры. Как правило, файлы хранятся на магнитных носителях внешних запоминающих устройств.

Администратор системы — это человек, ответственный за обеспечение ежедневного нормального функционирования системы. Он обязан запустить систему и проследить за завершением ее работы, регистрировать новых пользователей и вычеркивать выбывших, обеспечивать сохранность файловой системы и ее восстановление в случае повреждений, производить запуск программ, сообщающих, кто и в каком количестве использует ресурсы ЭВМ.

Дисциплина распределения времени про-

магнитных дисках (дискетах). Мы будем использовать накопители с дискетами диаметром 203 мм (8 дюймов) ГМД-7012 и дискетами диаметром 133 мм (5,25 дюйма) НГМД-6022.

Печать. Чтобы получать копию текстов на бумаге, в состав комплекса нужно включить печатающее устройство. Для малых систем удобно использовать термопечатающее устройство 15ВВ1780-002. Печать на нем производится нагреванием иголок, расположенных около бумаги, которая меняет цвет при повышении температуры.

Соберем и второй вариант этой системы, обладающий существенно более широкими возможностями.

В качестве процессора возьмем Электронику-100/25, имеющую более широкий набор команд, чем НМС1201, а также более высокое быстродействие. Кроме того, у нее есть так называемая расширенная (дополнительная) память.

Чтобы в дальнейшем показать различия в терминалах Видеотон VDT52130 и 15ИЕ-00-013, в данном варианте будем использовать VDT52130.

К Электронике-100/25 можно подключать большой набор внешних запоминающих устройств с существенно лучшими характеристиками. Используем следующие устройства:

накопитель ГМД-7012 для переноса программ на малую систему (ДВК-2М);

накопитель на жестких магнитных дисках СМ5400. Это основная быстродействующая память большого объема. Жесткий диск очень похож на большую грампластинку диаметром 356 мм. Он сделан из алюминия и покрыт магнитным лаком, как магнитофонная лента. Для защиты от пыли диск помещен в пластмассовую кассету с узкой щелью для ввода магнитных головок, отчего и называют его диск-кассетой;

накопитель ЕС-5061 на жестких дисках-пакетах (быстрая память очень большого объема). Диск-пакет похож на диск-кассету, но в пакете на одну ось надето несколько (до 20) жестких дисков, помещенных в съемный жесткий чехол;

накопитель на магнитной ленте СМ5300.01. Это медленное, но надежное запоминающее устройство большой емкости, напоминающее бытовой магнитофон, но имеющий не две и не четыре дорожки, а целых девять.

Для вывода информации на бумажную ленту будем использовать перфоратор ПЛ-150, для ввода — фотосчитыватель FS-1501.

В качестве печатающего устройства используем матричное печатающее устройство ударного действия DZM-180 или ROBOTRON-1156 (они отличаются друг от друга только габаритами и скоростью печати). Печатают они ударами иголок через красящую ленту. Иголок несколько, они образуют прямоугольник из точек — матрицу. От того, какие точки были «ударены», зависит, какой символ появится на бумаге.

Итак, у нас есть две системы — малая и средняя, на базе которых мы будем изучать программирование. Необходимо хорошо понимать, что состав аппаратуры в системе диктуется потребностями пользователя при решении конкретной задачи и может меняться и в количественном, и в качественном отношении. Мы берем указанный набор аппаратуры для того, чтобы рассмотреть возможно большее количество вопросов системного программирования. Обычно малую систему называют ДВК (диалоговый вычисли-

цессора — алгоритм, «решающий», какое количество времени процессор будет обслуживать каждую из введенных в ЭВМ программ (задач). Он может быть, например, таким: «Программы с высшим приоритетом (системные) получают время в тот момент, когда они его запрашивают; остальные программы получают по 0,1 с каждая в том порядке, в котором их вводили в ЭВМ; если все программы обслужены процессором, обслуживание начинается снова — с первой из имеющихся программ».

Архитектурой называют принципы, положенные в основу модели ЭВМ; одна машина, например, восьмиразрядная, другая — шестнадцатиразрядная; одна может считывать из своей памяти информацию по одному биту, другая — только байтами (порциями по 8 битов).

Производительность машины — это не

тельный комплекс), среднюю — «сотка» (по названию процессора Электроника 100/25) — так поступим и мы.

1.3. Как общаться с ЭВМ, или Понятие файла

Общение — это процесс обмена информацией между людьми, животными, человеком и ЭВМ. В каждый момент времени информация передается в каком-нибудь одном направлении от источника к приемнику. При этом важен не способ, с помощью которого эта передача происходит, а сам факт того, что приемник получил некоторую информацию, именно ту, которую передавал источник, и понял ее так, как этого хотел источник. Передача информации может происходить не только как перемещение ее от одного места к другому, но и как «перемещение» от одного момента времени к другому (хранение информации). Передача может происходить с изменением по определенным правилам (преобразование информации).

Таким образом, можно сказать, что главная задача процессора в системе состоит в общении с аппаратурой и человеком, в передаче информации от одного устройства к другому, от устройства к человеку, от человека к устройству. И, наконец, от одного момента времени к другому.

Но прежде чем передавать, нам необходимо определить, какими порциями можно передавать информацию в системе. Адресуемым элементом данных мы назовем минимальный элемент данных, к которому можно обратиться (прочитать или записать).

В нашей системе адресуемым элементом данных является блок, содержащий 256 шестнадцатиразрядных двоичных слов. Но пользоваться такими мелкими порциями информации неудобно, обычно используют более крупные — файлы.

Файл — это совокупность адресуемых элементов данных и способа их организации.

Файл может содержать произвольное количество блоков, находиться на любом устройстве. С файлом можно выполнять некоторый набор операций, который задается при проектировании системы. Ими могут быть: поиск файла по имени, создание записи о файле в каталоге (т. е. открытие его для записи информации), удаление записи о файле, переименование файла, чтение информации из файла, запись информации в файл и др.

Прежде чем проделать какую-либо операцию с файлом, мы должны определить, где он находится. Как у человека есть пол, имя и адрес, так и у файла есть спецификация. В общем виде она состоит из имени тома, где файл находится, имени и типа файла. Том — это машинный носитель информации, содержащий файлы. Так как он устанавливается на какое-то физическое устройство, то иногда в качестве имени тома используется имя устройства.

Для того чтобы найти файл на устройстве, необходимо где-то хранить информацию о файлах. Это нечто вроде оглавления книги, которое записано на томе и называется каталогом (справочником).

А теперь обратите внимание: если мы записали файл на магнитный диск, то можем через какое-то время найти файл на устройстве в автоматическом режиме, используя некоторую программу. Если мы запишем файл

только быстродействие. Производительность зависит от объема памяти — оперативной и внешней, от архитектуры, набора и качества периферийных устройств и многих других параметров.

Байт — 8 битов — зачастую используется в качестве единицы измерения памяти, так как его достаточно для кодирования любого символа (специального, цифры, буквы). Поскольку основная система счисления в ЭВМ двоичная, производные от него единицы являются степенями двойки: 1K байт=2^{10} байта=1024 байта, 1M байт=2^{20}=1 048 576 байта. Иначе говоря, в 8K байт можно записать примерно четыре страницы машинописного текста, а в 4M байт — том энциклопедии.

Задачи реального времени — это задачи, в которых время реакции системы на запрос пользователя меньше времени реакции самого пользователя.

на устройство печати, то информация останется на бумаге, ЭВМ прочесть ее не сможет. Если мы запишем файл на бумажную ленту, то прочитать ее можно, но для этого оператор должен ее найти и поставить на считывающее устройство. Отсюда видно, что все периферийные устройства делятся на два вида:

те, которые имеют справочник, где можно найти в автоматическом режиме запись об интересующем нас файле и затем прочитать его, называются устройствами файловой структуры;

те, которые не имеют справочника и не позволяют найти файл в автоматическом режиме, называются устройствами нефайловой структуры.

Спецификация файла на устройстве нефайловой структуры может состоять только из имени устройства. Имя и тип файла остаются неопределенными, и их определение не имеет смысла (например, для устройства печати).

Магнитная лента не имеет справочника как такового, но поскольку имена и типы файлов записаны на ленте и можно осуществлять автоматический поиск, то можно написать программу (и она имеется), которая собирает справочник в удобной для нас форме; поэтому накопитель на магнитной ленте является устройством файловой структуры.

При записи на магнитную ленту файлы размещаются один за другим, в порядке поступления. Записывать их в произвольном порядке (допустим, в середину ленты, а потом в начало) невозможно по техническим причинам. Поиск файла на ленте тоже ведется последовательным перебором.

При записи на магнитный диск мы можем записать файл на любое свободное место, а также прочитать произвольный файл, так как справочник содержит информацию не только о файлах, но и о свободных зонах.

Таким образом, логично назвать запоминающие устройства типа магнитной ленты устройствами с последовательным доступом, а типа магнитных дисков — устройствами с произвольным доступом.

Для устройств с произвольным доступом существует понятие состояния файла: он может быть пустым, временным и постоянным. Это очень важно для понимания того, что будет происходить в системе.

Допустим, мы захотели создать на устройстве файл и записать в него некоторую информацию. Как это происходит?

В каталог устройства заносится запись о том, что в таком-то месте тома зарезервировано указанное число блоков для файла с указанным именем и типом; файла еще не существует, а запись о нем в каталоге уже есть. Признак состояния такого файла — «временный». Говорят, что он открыт для записи. В общем случае на устройстве может существовать несколько временных файлов с одним именем и типом.

По мере поступления информации она записывается в зарезервированную область на томе и на устройстве начинает существовать настоящий файл, содержащий некоторую информацию. Признак его состояния — «временный».

После окончания работы с файлом мы можем его закрыть (сказать системе, что мы закончили работу с файлом и хотим его сохранить) и тем самым сделать постоянным. На одном устройстве должен быть только один файл, имеющий признак состояния «постоянный» и указанные имя и тип. Это значит, что в каталоге устройства должна быть только одна запись о файле с указанием имени и типа и признаком состояния «постоянный». В запись о файле с таким же именем и типом, если он существует, система занесет признак « пустой».

Если после окончания работы с файлом мы его не закроем (или, как говорят, освободим), то система занесет в запись о нем признак состояния «пустой». Имя и тип файла в дальнейшем использоваться не будут; система будет считать, что на месте, где находится файл (а он существует), имеется свободная (неиспользуемая) зона, равная ему по размеру.

Важно отчетливо понимать, что на устройстве с произвольным доступом

существует файл как объект, содержащий информацию, и запись о файле в каталоге устройства. В общем случае они могут быть не связаны между собой; мы можем производить какие-либо операции с файлом только тогда, когда сам файл и запись о нем в каталоге имеют однозначное соответствие и признак состояния файла — «постоянный».

Все, что мы говорили о файлах и их существовании, весь комплекс понятий о файлах носит название файловой системы. Более точно можно определить файловую систему как набор правил, позволяющий однозначно определить все операции над файлами. Эти правила будут определяться и уточняться на протяжении всей статьи.

Файл может содержать коды символов печатающего устройства, его можно вывести на печать и получить на бумаге текст. Такой файл мы назовем текстовым. Файл, содержащий двоичные коды, мы назовем двоичным. Если в нем коды записаны в соответствии с определенными правилами, то он может являться программой для процессора, его можно записать в оперативную память и выполнить. В общем случае двоичный файл может не быть программой. Некоторые файлы могут содержать часть произвольных двоичных кодов и часть символьных кодов.

1.4. Кто же правит в мире этом? или Базовый состав операционной системы

Как мы видели, мир системы может быть весьма разнообразным. Если каждый пользователь будет сам писать программы для управления многочисленными периферийными устройствами и файлами, то получится полная неразбериха и обмен файлами между пользователями будет практически невозможен. Поэтому лучше иметь единую управляющую программу или набор программ, которые называются операционной системой.

Операционная система — организованный набор программ, предназначенный для управления операциями ввода-вывода информации и программами пользователя, а также для облегчения разработки программ и файлов пользователем.

Мы выделяем программы из общего множества файлов; хотя первоначально пользователь являлся программистом и создавал программы, теперь это не обязательно. Его основная работа может заключаться в создании текстов (текстовых файлов) или управлении производством с помощью информационной системы.

Программа пользователя не обязательно должна быть написана пользователем. Это может быть программа, входящая в состав операционной системы, но служащая для удовлетворения его нужд.

Мы будем говорить об операционной системе (ОС) RT-11 (Real Time systems PDP-11), разработанной фирмой DEC, прежде всего потому, что она явилась родоначальницей всех однопользовательских ОС для малых ЭВМ. Различные же ОС, совместимые с RT-11, внесли мало нового и, как правило, плохо документированы.

Соответствие различных версий RT-11 и других ОС

RT11-V2.0	FOBOS, ФОДОС-2В
RT11-V3.0	FOBOS, РАФОС, ОСДВК
RT11-V4.0	РАФОС-2, ОСДВК
RT11-V5.0	ФОДОС-II
RT11-V5.1	ФОДОС-III

В 1972 г. предполагалось, что RT-11 будет маленькой, быстродействующей, простой в обращении ОС для семейства компьютеров PDP-11. Она развивалась как однопользовательская система для задач реального времени и вычислений; ее основными приложениями были получение данных, управление технологическими процессами или научными экспериментами и, конечно, разработка программ, расширяющих возможности системы.

В настоящее время RT-11 представляет собой мощную, многозадачную, многотерминальную, многомашинную однопользовательскую операционную систему. На базе RT-11 в нашей стране и за рубежом разработаны совместимые с ней операционные системы.

Приводимые примеры будут ориентированы на использование RT11-V5.1, но не все объяснения справедливы для всех перечисленных операционных систем. Любые отличия между версиями будут оговариваться в каждом конкретном случае.

1.4.1. На золотом крыльце сидели, или Структура операционной системы

Продолжая наши рассуждения, мы попробуем собрать операционную систему из кирпичиков — программ. Будем исходить только из того определения операционной системы, которое было дано выше, и из представлений о потребностях пользователя.

Начнем читать определение.

1.4.1.1. ...Царь, царевич, или Системные программы

Операционная система должна управлять операциями ввода-вывода.

Проанализируем это условие. Ввод или вывод информации происходит через какое-либо устройство. Оно должно быть определено в системе. Каждое устройство имеет свою конструкцию, свои специфические правила обмена информацией, следовательно, для обмена информацией с каждым устройством должна быть своя программа. Эти программы называются драйверами устройств (от drive — приводить в движение) и являются системными. Если мы хотим ввести в систему новое устройство, то прежде всего нужно иметь его драйвер, программу, которая обеспечивает обмен информацией с этим устройством в виде, принятом в ОС.

В RT-11 драйвер получает имя соответствующего устройства в системе. Ниже приводятся системные имена устройств и драйверов, которые мы будем использовать.

Часть драйверов общие для ДВК и «сотки». Это:

TT — драйвер системного терминала; обеспечивает ввод информации с клавиатуры терминала и вывод на экран дисплея; устройство с последовательным вводом-выводом нефайловой структуры;

DX — драйвер накопителя на гибких магнитных дисках ГМД-70 или ГМД-7012; устройство с произвольным доступом файловой структуры;

LP — драйвер печатающего устройства DZM-180 или ROBOTRON-1156 (1152); вывод информации подобен выводу на экран дисплея, но отсутствуют многие специальные функции терминала; устройство последовательного вывода нефайловой структуры;

SL — драйвер редактора командной строки. Физически такого устройства не существует; драйвер появился только в RT11-V5.0 и позволяет редактировать вводимую с терминала или уже введенную, но предназначенную для повторного ввода строку;

NL — еще одно фиктивное устройство, не существующее физически. Оно «делает вид», что выводит информацию на устройство нефайловой структуры, однако физически никакого вывода не происходит.

В ДВК используется драйвер накопителя на гибких магнитных дисках НГМД-6022, который имеет имя MX. Он разработан в СССР и не входит в исходную систему RT-11. Этот накопитель нельзя подключить к «сотке».

Для «сотки» добавляются драйверы устройств, отсутствующих в ДВК:

RK — драйвер накопителя на диск-кассетах RK05 или СМ5400; устройство с произвольным доступом файловой структуры;

DP — драйвер накопителя на дисках-пакетах RP03 или ЕС-5061 емкостью 20М байт; устройство с произвольным доступом файловой структуры;

MT — драйвер накопителя на магнитной ленте СМ5300 или TM11; устройство с последовательным доступом файловой структуры;

VM — драйвер виртуальной памяти. Программа, позволяющая использовать дополнительную память, которая имеется в «сотке», как сверхскоростное устройство файловой структуры с произвольным доступом. Драйвер появился только в RT11-V5.0;

LD — драйвер логического диска. Физически такого устройства, как это видно из названия, нет, однако эта программа позволяет на любом физическом

диске (точнее, на устройстве файловой структуры с произвольным доступом) работать с конкретным двоичным файлом как с устройством файловой структуры с произвольным доступом. Использование того драйвера в ДВК возможно, но не эффективно. LD появился в RT11-V5.0 как логическое развитие имевшихся в предыдущих версиях драйверов UX, UY, UZ. Чтобы понять, зачем он нужен, достаточно представить себе, как неудобно пользоваться очень толстой книгой и насколько проще читать ее же, напечатанную в нескольких томах, беря в руки том за томом. Когда мы начнем работать с накопителями типа RK или DP, мы в этом убедимся.

Итак, мы подобрали для обеих систем наборы программ для ввода-вывода. Однако для организации всех операций обмена информацией одних драйверов недостаточно. Необходима еще и некоторая программа, которая бы ими управляла и, в совокупности с ними, составляла бы систему ввода-вывода. Такая программа есть в RT-11 и, несмотря на то что ее функции мы будем рассматривать отдельно, в самостоятельном виде она не применяется. Программа называется USR (User Service Routine — программа обслуживания пользователя) и занимается организацией всех операций ввода-вывода — как тех, которые мы захотели выполнить в диалоговом режиме командами с клавиатуры, так и тех, которые требуется выполнить программе пользователя, исполняющейся в данный момент.

По определению, операционная система должна управлять программами пользователя, т. е. загружать программы в память, следить за распределением памяти, за событиями в системе, вести учет времени и выполнять много других вспомогательных функций, облегчающих разработку программ и файлов. Для этого необходима специальная управляющая программа — монитор. Она управляет всеми другими программами и поэтому должна постоянно находиться в оперативной памяти или, как говорят, быть резидентной в памяти. Из совокупности этих посылок и вытекает название управляющей программы в RT-11 RMON (Residented MONitor).

Так как программы ОС хранятся на машинном носителе (магнитной ленте или диске), необходима начальная программа, позволяющая загрузить монитор ОС в оперативную память процессора и запустить его. Он называется BSTRAP (bootstrap — загрузчик). Загрузчиков у системы несколько, работают они в разное время и располагаются в разных программных модулях. BSTRAP располагается в файле монитора.

Для передачи системе информации от человека необходима программа обработки команд с клавиатуры. В RT-11 она состоит из клавиатурного монитора KMON (Keyboard MONitor). Он выполнен как самостоятельная программа и работает под управлением резидентного монитора и интерпретатора командной строки (Comand String Interpretator-CSI) (он реализован как часть программы USR, чтобы облегчить обращение к системе из программы пользователя).

USR, RMON, BSTRAP и KMON хранятся в виде единого файла, имеющего имя RT11¤¤, где «¤¤» обозначает тип монитора (BL, SJ, FB, XM, TS, AI). Про типы мониторов мы будем говорить в дальнейшем. А пока используем монитор RT11SJ (Single Job — одно задание), который позволяет запускать одновременно только одну программу. Все, что будет сказано о нем без оговорок, справедливо и для остальных.

В RT-11 V2, V3 имя файла монитора другое — *CC*MN ¤ ¤, где *CC* — двухбуквенное имя устройства, которое будет использоваться как системное, ¤¤ — тип монитора. Например, в ОС ДВК используется монитор MXMNSJ.

Конечно, можно работать в системе и не зная этих подробностей. Однако поиск решения в тупиковых ситуациях основывается на понимании описанных мелочей. Это во-первых. А во-вторых — в другой системе программы будут иметь другие имена, будут по-другому делить между собой функции — но и функции, и программы останутся.

Все описанные программы и файлы самостоятельно работать не могут, составляют ядро операционной системы и имеют тип .SYS..

1.4.1.2. ...Сапожник, портной, или Вспомогательные программы

Вспомогательные программы, или утилиты (utility), расширяют функциональные возможности операционной системы. Их набор определяется потребностями пользователя при решении конкретной задачи, однако всегда есть нужда в программах, позволяющих уменьшить ущерб от сбоев и неполадок системы.

Одна из самых важных функций — копирование файлов. Оно выполняется при переносе файла с устройства на устройство, при выводе на печать. Скопировать файлы операционной системы с эталона — первое, с чем сталкивается пользователь. В RT-11 для этого используется программа PIP (Peripheral Intercharge Programm — программа обмена с периферией).

Файлы хранятся на устройстве внешней памяти файловой структуры, которое имеет каталог. С ним надо работать. Организацию каталога, внесение в него записей, идентификатора тома и имени владельца, копирование информации с устройства на устройство с использованием абсолютных номеров блоков, а не имен файлов, выполняет программа DUP (Device Utility Programm — программа обслуживания устройств).

Программист работает вслепую. Никто не видел файл, хранящийся на магнитном диске, и когда возникают сбои — ошибки аппаратуры или программ, ошибки самого пользователя — очень хочется посмотреть, что произошло в системе, что изменилось, получить хоть малейшую информацию для анализа возникшей ситуации.

Одной из служащих для этого програм является DIR (directory). Она позволяет распечатать справочник устройства файловой структуры в удобном для пользователя виде. Используя информацию программы, полученную в различные моменты времени, можно сказать, что происходило в системе и что не происходило.

DIR выдает пользователю информацию о файлах и о занимаемых ими блоках тома. Часто требуются более подробные сведения, например, какие коды находятся в данном месте тома. Для этого в RT-11 используется программа DUMP. Следует отметить, что программа DUMP часто позволяет получить информацию в тех случаях, когда другие программы бессильны.

Итак, в качестве вспомогательных программ мы включили в операционную систему PIP и DUP — это «руки» ОС, DIR и DUMP — это ее «глаза».

1.4.1.3. ...Кто ты будешь такой? или Остальные программы ОС

Рассмотренные программы используются в большинстве случаев, другие включаются в состав ОС только для расширения набора функций системы. Это могут быть программы, поставляемые вместе с операционной системой или написанные пользователем. Чтобы они могли входить в состав операционной системы, их следует писать с учетом правил обмена информацией и правил существования системы. Они могут несколько меняться от версии к версии, поэтому для надежной работы системы все программы должны быть одной версии.

1.5. Указ становится законом, или Полное определение системы

Теперь мы можем, наконец, определить то, что будем называть системой. Это организованная совокупность набора аппаратуры (аппаратная часть системы — hardware), набора программ (операционная система — software) и правил обмена информацией (interface), предназначенная для удовлетворения нужд пользователя.

Предлагаемая литература должна не только помочь читателю разобраться в данной главе, но и подготовить к чтению следующих глав.

Варга Б., Димень Ю., Лопариц Э. Язык, музыка, математика / Пер. с венг. Ю. А. Данилова. М.: Мир, 1981.

Эта книга не имеет никакого отношения ни к программированию, ни к вы-

числительной технике, ни к операционным системам. Даже к содержанию этой главы она имеет весьма косвенное отношение. Для чего же ее предлагается прочитать именно сейчас? — Чтобы сформировать у читателя умение находить и желание искать аналогии в окружающем нас мире. Без этого дальнейшая работа в системе будет неэффективной и не продвинется дальше любительства.

Рафаэл Б. Думающий компьютер / Пер. с англ. В. П. Стефаника. М.: Мир, 1979.

Несмотря на то что Бертран Рафаэл специалист в области искусственного интеллекта, в предлагаемой книге должно привлечь внимание удивительно точное описание сущности ЭВМ и значение, которое Рафаэл придает постановке задачи. В точной постановке содержится более половины решения.

Колин А. Введение в операционные системы / Пер. с англ. И. Х. Зусман. М.: Мир, 1975.

Одна из самых тоненьких, она с прекрасным юмором рассказывает об основных чертах операционных систем, не опираясь на какую-нибудь конкретную систему. Эпиграфы к каждой главе книги настолько точно формулируют идею, что глава является скорее иллюстрацией к эпиграфу.

Уорвик М. Введение в операционные системы. Основные понятия // Супервизоры и операционные системы / Под ред. Дж. Каттла и П. Робинсона. М.: Мир, 1972.

Продолжение следует

М. РЯЗАНСКИЙ, О. ЯЦЮК

Палитра компьютерной графики

Айвэн Сазерленд — пионер применения компьютеров для построения и обработки изображений — как-то заметил: «Дисплей, подключенный к ЭВМ, представляется мне окном в Страну чудес Алисы, где программист может изображать либо объекты, подчиняющиеся хорошо известным законам природы, либо воображаемые объекты, подчиняющиеся законам, описанным в программе. С помощью дисплея я сажал самолет на палубу движущегося авианосца, следил за движением элементарной частицы в потенциальной яме, летал на ракете с околосветовой скоростью и наблюдал за таинствами внутренней жизни ЭВМ».

Действительно, с помощью ЭВМ сейчас можно получать удивительные изображения, которые не получишь никаким другим способом. Стала реальностью ЭВМ, рисующая мультфильмы, но чего ей это стоит? Для CRAY X—MP, выполняющей 200 млн. операций над числами с плавающей запятой в секунду (200 Мегафлоп), необходим целый месяц работы, чтобы сделать одну серию «Ну, погоди!». Однако такие машины в каждую школу не поставишь, слишком дорого, да и программы для них пишут большие коллективы программистов. Школьникам это не под силу. А как же быть? Что можно сделать на школьных компьютерах?

Мы учим черепашку

В конце 60-х гг. Сеймур Пейперт и его коллеги из Массачусетского технологического института предложили язык для программирования механических рисующих устройств — графопостроителей. Язык назвали LOGO. Идея его настолько проста, что позволяет обучать этому языку даже первоклассника.

Пусть у нас есть дрессированная черепашка, которая умеет ползать по песку, и если она опустит хвост, то на песке останется след. Так как черепашка не столь умна, как человек, то и команд она понимает мало. В самом упрощенном варианте команды могут быть следующие:

опусти хвост;
поднимни хвост;
ползи вперед N шагов;
повернись на L градусов.

Очевидно, что черепашка в один момент времени может выполнять только одну команду, а это значит, что повернуться она может только на месте. Направление поворота зависит от того, с каким знаком мы укажем число градусов. Примем, например, что если число градусов для поворота указано со знаком минус, то черепашка должна повернуться налево, иначе — направо. Примитивность такого языка не должна пугать — ведь чем меньше команд, тем легче их запомнить.

А возможности? Возможности оказываются достаточно богатыми. Попробуем, для примера, заставить черепашку нарисовать своим хвостом пятиконечную звезду с расстояниями между концами, например, 50 шагов. Необходимые для этого команды в нужной последовательности запишем в столбик:

```
опусти хвост
  ползи вперед 50 шагов
    повернись на 144 градуса
  ползи вперед 50 шагов
    повернись на 144 градуса
  ползи вперед 50 шагов
    повернись на 144 градуса
  ползи вперед 50 шагов
    повернись на 144 градуса
  ползи вперед 50 шагов
    повернись на 144 градуса
подними хвост
```

Как видим, используя обычные разговорные команды, мы заставили нашу дрессированную черепашку нарисовать звезду (кто не верит, может сам повторить все действия с карандашом). Правда, мы пять раз повторили группу из двух команд; добавив в язык команду «повтори N раз», мы сократим размер нашей программы и уменьшим возможность появления ошибки:

```
опусти хвост
  повтори 5 раз [ползи вперед 50 шагов;
                 повернись на 144 градуса]
подними хвост
```

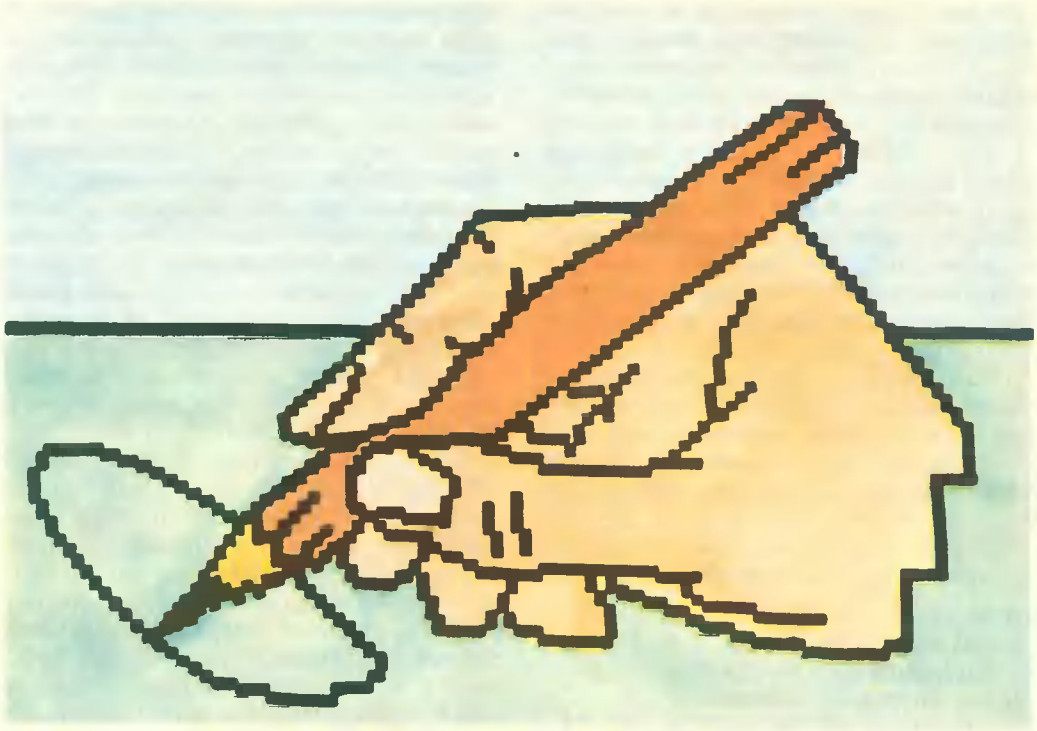
Из приведенного выше примера можно видеть, что LOGO обладает сходством с естественным языком. Язык LOGO имеет словарь, содержащий слова, числа и другие символы (все вместе называемые знаками), которые можно связать друг с другом, чтобы получить более крупные конструкции, подобные предложениям естественного языка. В исходной версии LOGO команды выражались на английском языке, но так как язык близок к естественному, то не составляет трудности перевести их на русский. Современная версия включает значительно большее число команд, чем мы здесь привели, что позволяет писать сложные программы. На LOGO можно, например, описать построение чертежа сложной детали механизма и заставить машину рисовать эту деталь на любом чертеже, где это потребуется.

Аналогия с черепашкой позволяет объяснить «на пальцах», как пользоваться этим языком управления механическим рисующим устройством, любому ребенку. Достоинство такого подхода еще и в активизации деятельности ребенка, который сам «учит» черепашку рисовать, активен в этом процессе. Он пишет программу, не зная страшного слова «программирование», потому что его «программирование» заключается в выражении своих желаний: ползи, повернись, рисуй. В результате язык LOGO стал применяться для обучения программированию. Школы, в которых установлены КУВТ «Ямаха», могут активно его использовать.

От гвоздика до гвоздика ниточку протянем

А что же делать в школах, в которых установлены КУВТ на базе микро-ЭВМ БК-0010?

Принципы построения изображения в языке Фокал для БК-0010 совершенно другие. Подходящей аналогией здесь может служить прямоугольная доска, все поле которой разделено на клетки. В середину каждой клетки можно вбить гвоздик, если нужно отметить точку, и вынуть гвоздик, если точку нужно стереть. Если один гвоздик вбит, то от него можно протянуть нитку до другой клетки, в середину которой вбивается гвоздик и на нем закрепляется нитка. Таким образом, при рисовании линий движение как процесс остается, но это движение от точки к точке, а не только вперед.



При этом надо заметить, что координаты следующей точки необходимо вычислять. В отличие от метода черепашки, который можно назвать векторным, метод с гвоздиками на доске можно назвать растровым, так как доска с клетками образует растр.

Главное отличие растрового метода от векторного в том, что, имея растр, мы можем нарисовать точку. Чтобы нарисовать ее в векторном методе, нам надо заставить черепашку проползти хотя бы на малое расстояние, т. е. нарисовать линию малой длины. В растре явление обратное — мы рисуем линию как совокупность точек (прекрасная иллюстрация к определению линии как геометрического места точек).

Графические средства языка Фокал достаточно просты. Мы имеем два оператора, с помощью которых можем нарисовать любой рисунок:

X FT (КОП, X, Y) — он позволяет установить луч дисплея на точку с координатами X и Y. Код операции КОП может иметь два значения: «1» — высветить точку (вбить гвоздик), «0» — погасить точку (вынуть гвоздик);

X FV (КОП, X, Y) — он позволяет провести (или стереть, в зависимости от КОП) линию (натянуть нитку) от точки, где находится луч, до новой точки, определяемой координатами X, Y.

Попробуем нарисовать на экране дисплея пятиконечную звезду. Графические средства у нас более сложные, так что, во-первых, нам придется написать настоящую программу для интерпретатора языка Фокал; во-вторых, так как мы уже не на бесконечном поле, а на ограниченной доске размером 512 клеток по горизонтали и 256 клеток по вертикали, нам придется несколько подготовиться, а именно: очистить доску от старых «гвоздиков»; указать координаты клетки, с которой мы начнем рисовать; установить луч на точку с этими координатами; определить направление, в котором мы начнем рисовать. Необходимо также отметить, что координаты каждой следующей точки нам придется вычислять. Вот что получится в итоге.

```
1.01 S X=100; S Y=100
1.02 X FCHR(12)
1.03 X FT(0, X, Y)
1.10 S A=0
```

```
1.11  S X=X+50*FCOS(A);
      S Y=Y+50*FSIN(A)
1.12  X FV(1, X, Y)
1.20  S A=A+144
1.21  S X=X+50*FCOS(A);
      S Y=Y+50*FSIN(A)
1.22  X FV(1, X, Y)
1.30  S A=A+144
1.31  X=X+50*FCOS(A);
      S Y=Y+50*FSIN(A)
1.32  X FV(1, X, Y)
1.40  S A=A+144
1.41  S X=X+50*FCOS(A);
      S Y=Y+50*FSIN(A)
1.42  X FV (1, X, Y)
1.50  S A=A+144
1.51  S X=X+50*FCOS(A);
      S Y=Y+50*FSIN(A)
1.52  X FV(1, X, Y)
```

Получилось довольно сложно. Как написать такую программу, уже не объяснишь первокласснику. Несмотря на то что мы «тянули ниточку от гвоздика к гвоздику», нам потребовались и подготовка исходного положения, и знание математики, и вычисления, часть которых черепашка делала не задумываясь, а остальные ей просто были не нужны. Хотя и здесь мы можем сократить программу, использовав повторение группы команд:

```
1.01 S X=100; S Y=100; S A=—144
1.02 X FCNR(12)
1.03 X FT(O, X, Y)
1.04 F I=1,5; D 2
1.05 Q
2.01 S A=A+144
2.02 S X=X+50*FCOS(A);
     S Y=Y+50*FSIN(A)
2.03 X FV (1, X, Y)
```

Возникает резонный вопрос: если есть простой графический язык, то зачем нужен сложный? Ответ таков: попробуйте нарисовать эллипс с полуосями A и B на языках LOGO и Фокал.

На Фокале получится изящное решение:

```
1.01 S N=64
1.02 S PI=3.14159/N;
     S X=256; S Y=128
1.03 F I=0,2*N;
     X FV (1, X+A*FCOS(PI*I),
     Y+B*FSIN(PI*I))
```

А для решения этой задачи на LOGO придется немало поломать голову.

Применение графических средств Фокала

В отличие от LOGO, на котором мы можем рисовать только линии, графические средства Фокала, дополненные возможностью вычислений, превращают БК-0010 в прекрасное демонстрационное средство. Вспомним определения. Например, окружность — это геометрическое место точек, равноудаленных от точки, называемой центром окружности. Без особого труда мы можем рисовать на экране параболы и гиперболы, эллипсы и прямоугольники, любую кривую, которую описывает уравнение. График синусоиды может быть нарисован на экране с помощью следующего фрагмента программы:

```
1.01 A введите значение амплитуды; A
1.02 X FCHP(12)
1.03 X FT(0, 0, 128);
     X FV (1, 512, 128);
     X FT(0, 0, 128)
1.04 S PI=3.14159/32
1.10 F I=0,512;
     X FT(1, I, 128+A*FSIN(PI*A))
1.11 G 1.01
```

Он рисует в виде точек, пунктирной линией. Заменив один оператор, можно нарисовать график сплошной линией.

В качестве примера рассмотрим программу, которая позволяет ученику исследовать зависимость формы сложного периодического колебания от амплитуд его гармоник. Сама программа не приводится, так как занимает много места, но принципы, использованные при ее написании, приемлемы и в других случаях.

Работа происходит в диалоговом режиме. Сначала текст на экране информирует, что можно сделать и какие исходные данные необходимо вводить. После их ввода (амплитуды основного колебания и шести гармоник) на экране появляются оси координат и (в виде отдельных точек) графики гармоник в соответствии с заданными амплитудами. Затем на фоне пунктирных графиков сплошной линией строится результирующее сложное колебание. На все рисунки уходит 6 мин, затем программа переходит в режим ввода исходных данных.

Несмотря на ее простоту, даже у взрослых возникает интерес: а какая форма результирующего колебания будет, если амплитуду одной гармоники сделать побольше, а другой поменьше?

Разработка подобных программ не представляет трудностей для учителей школ. В настоящее время ведется разработка программы—исполнителя с русскими командами языка LOGO для БК-0010. Это будет попытка дать учителям школ как инструмент обучения дрессированную черепашку, которую даже ребенок может учить.

В заключение отметим, что принципы построения изображений в векторной и растровой форме применяются не только в компьютерной графике, но и в тех областях науки и техники, где нет традиции графического представления объектов и процессов. Например, для базы географических данных удобно вывод информации осуществлять в векторной форме. Однако, если мы хотим запомнить какое-нибудь изображение с детальными подробностями, например, кометы Галлея, нам удобнее воспользоваться растровым методом.

Компьютерная графика — область, ранее доступная только специалистам — становится привычной. Даже в детских садах работают с дисплеями, как с инструментом для рисования и, конечно же, для развития воображения. Мышление и программирование на языке графических образов становится неотъемлемой частью обучения.

Литература

Андриес Ван Дам. Программное обеспечение машинной графики // В мире науки. 1984. № 11; Современный компьютер. М.: Мир, 1986.

Лоуренс Г. Теслер. Языки программирования // В мире науки. 1984. № 11; Современный компьютер. М.: Мир, 1986.

Эмери М. Очередная революция в обучении? // Перспективы, вопросы образования. 1986. № 3.

Э. ИКАУНИЕКС

Основы языка Бейсик

10. Программа на Бейсике. Работа с программой

Программа на Бейсике состоит из нумерованных строк. В каждой строке можно записать одну или несколько команд. Если в одной строке записывается несколько команд, их надо разделить двоеточиями,

```
100 A=3: B=3: PRINT «A+B=»; A+B
```

Одна строка программы может занимать на экране более одной физической строки, содержащей 40 символов. Обычно максимально допустимая длина строки программы — 255 символов (такая же, как максимальная длина значения литерной переменной). Если при вводе длинной строки программы экранная строка уже заполнена, то текст автоматически переносится на следующую экранную строку (без какого-либо символа переноса), например,

```
1000 PRINT «ТЕКСТ РАЗРЫВАЕТСЯ НА ПРОИЗВОЛЬНОМ МЕСТЕ»
```

При выполнении этой команды ЭВМ выводит весь текст на одной физической строке:

```
ТЕКСТ РАЗРЫВАЕТСЯ НА ПРОИЗВОЛЬНОМ МЕСТЕ
```

Номера строк указывают на порядок выполнения команд. Кроме того, они позволяют указать требуемую строку (например, при внесении исправлений). По команде RUN программа выполняется начиная со строки с наименьшим номером и далее продолжая в порядке возрастания номеров, если только не встречается команда, которая специально меняет последовательность выполнения строк (см. ниже).

Номера строк — произвольные натуральные числа от 1 до 65529 (в некоторых версиях Бейсика максимальный допустимый номер строки гораздо меньше). Обычно строки нумеруют не числами 1, 2, 3..., а 10, 20, 30... . Это облегчает добавление новых строк между уже введенными и занумерованными.

В предыдущем пункте уже говорилось о том, как ввести программу в ЭВМ. Строки программы можно вводить в произвольном порядке (например, сначала сотую, а потом десятую) — при размещении в памяти они автоматически упорядочиваются.

Продолжение. Начало см. № 3 за 1986 г.

Если какую-то строку надо заменить, то достаточно ввести новый вариант строки с тем же номером. Если строку надо убрать, то достаточно ввести ее номер (т. е. пустую строку с номером исключаемой строки).

Текст программы, находящийся в памяти, можно просмотреть, дав команду

LIST

(LIST — перечислить, перелистать). Выполняя ее, ЭВМ показывает программу на экране строку за строкой.

Чтобы ЭВМ «забыла» программу, находящуюся в ее памяти, надо ввести команду

NEW

(NEW — новый). При этом память ЭВМ освобождается для ввода новой программы.

Рассмотрим пример (текст, выведенный ЭВМ, выделен жирным шрифтом).

```
NEW
10 PRINT «ВВЕДИТЕ СВОЕ ИМЯ!»
20 INPUT V
30 PRINT «РАД ПОЗНАКОМИТЬСЯ»
```

Прочитав внимательно текст программы, замечаем, что допущена ошибка. Переменная, в которой будет храниться введенное пользователем имя, должна быть литерного типа и ее имя должно кончаться на $/. Надо исправить строку 20:

```
20 INPUT V$
```

Вставим еще обращение по имени:

```
25 PRINT «ПРИВЕТ, «;V$ » !»
```

Отделим имя, введенное пользователем, от последующего текста:

```
22 PRINT
```

Просмотрим теперь всю программу:

```
LIST
10 PRINT «ВВЕДИТЕ СВОЕ ИМЯ!»
20 INPUT V$
22 PRINT
25 PRINT «ПРИВЕТ, «V$ » !»
30 PRINT «РАД ПОЗНАКОМИТЬСЯ»
Ok
```

Дадим команду выполнения этой программы:

```
RUN
ВВЕДИТЕ СВОЕ ИМЯ!
? НАТАЛЬЯ
ПРИВЕТ, НАТАЛЬЯ!
РАД ПОЗНАКОМИТЬСЯ
Ok
```

При выключении ЭВМ теряется все, что записано в ее оперативной памяти. Нашу программу можно сохранить, записав ее во внешнюю память (на дискету, диск или кассету). Это делается командой SAVE (SAVE — спасать, сохранять). На различных ЭВМ и для различных типов внешней памяти эта команда имеет различные варианты, например,

SAVE «⟨имя программы⟩»

Сохраним нашу программу под именем NAME:

```
SAVE «NAME»
```

Очистим оперативную память:

```
NEW
```

Проверим, что у нас в оперативной памяти:

```
LIST
Ok
```

Программы там уже нет: команда LIST выполнена — только нечего выводить на экран.

Если надо перевести программу из внешней памяти в оперативную, следует дать команду

LOAD «⟨имя программы⟩»

(LOAD — загрузить). Продолжая наш пример, введем

```
LOAD «NAME»
```

Возвратилась ли наша программа в оперативную память? Проверим:

```
RUN
ВВЕДИТЕ СВОЕ ИМЯ!
? НАТАША
ПРИВЕТ, НАТАША!
РАД ПОЗНАКОМИТЬСЯ
Ok
```

Выполнение команды LOAD не стирает запись программы во внешней памяти — она только копируется в оперативную.

Программы должны быть понятны не только ЭВМ, но и человеку. Их часто приходится исправлять или менять, а исправлять можно только понятное. Поэтому желательно в самой программе коротко комментировать ее назначение, алгоритм. Для этого используется команда

REM ⟨строка символов⟩

(REMark — заметка). Встретив REM, интерпретатор игнорирует все следующие за этим ключевым словом символы и переходит к выполнению следующей строки программы.

Каждую программу желательно начинать комментарием с объявлением имени программы, ее автора, даты разработки и другой необходимой информации. Так, например, к только что рассмотренной программе следовало бы вначале добавить комментарий

```
5 REM *** NAME ***
6 REM * ЗНАКОМСТВО ПО ИМЕНИ *
7 REM «IKAUNIEKS 28—FEB—86»
```

Вопросы

1. Сколько команд разрешается записывать в одной строке программы? (Сколько угодно; надо только отделять их друг от друга двоеточием и заботиться, чтобы общая длина строки не превышала 255 символов.)

2. Как нумеровать строки программы? (Можно любыми натуральными числами, удобно — числами 10, 20, 30... .)

3. Какие команды употребляются для
а) просмотра текста программы;
б) освобождения памяти для новой программы;
в) записи программы во внешнюю память;
г) загрузки программы из внешней памяти в оперативную?

(LIST; NEW; SAVE; LOAD.)

4. Для чего в программе нужны комментарии? (Только для того, чтобы человеку было понятно, что делает программа.)

Упражнения.

1. Написать программу (не более двух строк!), которая потребует от пользователя ввести число X и отпечатает целую часть этого числа.

Ответ. Возможен вариант

```
10 PRINT «ВВЕДИТЕ X», INPUT X
20 PRINT INT (X)
```

2. Вам нужно записать программу, находящуюся в оперативной памяти, во внешнюю память под именем PROQ2, потом вызвать и выполнить программу PROG1, записанную ранее во внешней памяти. Напишите соответствующие команды.

Ответ.

```
SAVE «PROG2»
LOAD «PROG1»
RUN
```

11. Команды конца работы и передачи управления

ЭВМ кончает работу над выполнением программы по команде

```
END
```

(END — конец). Выполняя ее, ЭВМ переходит из программного режима в командный и ожидает ввода пользователем следующей команды. В тексте команду можно записывать и несколько раз; с другой стороны, иногда можно обойтись и вовсе без этой команды: выполнив последнюю строку программы, ЭВМ останавливается сама — ей больше нечего делать, команды кончились.

Имеется вторая команда останова программы

```
STOP
```

(STOP — остановиться). Разница между END и STOP только в том, что STOP только приостанавливает работу по программе, после этой команды можно при желании работу продолжить, END же прекращает работу окончательно.

Если бы в программах можно было использовать только рассмотренные до сих пор команды, то каждую команду в программе ЭВМ выполнила бы только один раз, и на каждое действие нужно было бы записать свою команду. Так, например, для вывода таблицы кубов чисел 1, 2, ..., 500 нужно было бы выписать 500 строк

```
10 PRINT 1, 1^ 3
20 PRINT 2, 2^ 3
...
5000 PRINT 500, 500^3
```

Однако в Бейсике имеются средства, позволяющие такие программы записать очень коротко.

В этом разделе рассмотрим команду GOTO (GO TO — идти к), с помощью которой можно менять естественную последовательность выполнения строк программы. Ее общий вид

GOTO ⟨номер строки⟩

В результате ее выполнения следующей выполняемой командой будет та, чей номер указан за ключевым словом GOTO (программисты говорят, что управление передается строке с указанным номером), например,

```
NEW
10 PRINT «2*2=4»
20 GOTO 10
RUN
```

Эта программа никогда не закончит свою работу, так как после вывода текста 2*2=4 команда GOTO 10 возвращает к строке 10, которая опять выводит «2*2=4» и т. д. Пользователь, разумеется, может вмешаться и остановить такую «вечно» работающую программу (ее называют «зациклившейся») — для этого на каждой ЭВМ имеется специальная клавиша.

Командами GOTO следует пользоваться как можно реже, так как они делают программу трудно читаемой для человека — приходится особо следить за тем, в каком порядке выполняются строки программы. Недаром в алгоритмический язык эта конструкция не включена вообще. В рассматриваемом нами подмножестве Бейсика, к сожалению, обойтись совсем без GOTO невозможно.

Вопросы.

1. В чем разница между END и STOP? (END заканчивает выполнение программы, STOP только приостанавливает.)

2. Почему следует избегать команду GOTO? (Программы с многими GOTO трудно понять.)

Упражнения.

1. Дана плохая программа

```
10 INPUT X
20 GOTO 40
30 PRINT X+Y : GOTO 60
40 INPUT Y
50 GOTO 30
60 END
```

Сделайте ее по возможности короткой.

Ответ.

```
10 INPUT X, Y
20 PRINT X+Y
```

2. Что произойдет, если пользователь введет строки

```
NEW
10 X=1
20 X=2*X
30 PRINT X
40 GOTO 20
```

Ответ. Программа будет печатать степени двойки 2, 4, 6, 8, 16... до тех пор, пока очередное значение X не станет слишком большим — тогда произойдет переполнение и программа закончит работу.

12. Команда ветвления

Рассмотрим теперь команду ветвления, которая на алгоритмическом языке записывалась с помощью служебных слов **если, то, иначе, все.** Команда ветвления на Бейсике по существу та же — только записывается по другим правилам. Служебные слова — на английском языке: IF (если), THEN (то), ELSE (иначе). Служебное слово все не нужно, так как вся команда ветвления должна разместиться на одной строке программы и конец строки — это конец команды ветвления.

Ее общий вид

IF ⟨условие⟩ THEN ⟨серия1⟩ ELSE ⟨серия2⟩

Выполняя ее, ЭВМ проверяет соблюдение условия. Если оно соблюдено, выполняется серия1, если нет — серия2.

Таким образом, в зависимости от соблюдения условия выполняется одна и только одна из двух серий. Каждая серия может состоять из одной или нескольких команд.

В некоторых версиях Бейсика разрешается только сокращенная форма команды ветвления

IF ⟨условие⟩ THEN ⟨серия⟩

В этом случае при соблюдении условия серия команд выполняется, а при несоблюдении пропускается — управление сразу передается на следующую строку программы.

Отметим, что сокращенной формой команды ветвления вполне можно обойтись. Так, например, вместо строки

```
100 IF X>0 THEN Y=X*X : Z==1 ELSEY=X+1 : Z=-1
```

можно записать

```
100 IF X>0 THEN Y=X*X : Z=1
105 IF X<=0 THEN Y=X+1 : Z=-1
```

В общем случае в первой команде проверяются соблюдение данного условия, в другой — противоположного ему. Впредь будем пользоваться преимущественно сокращенной формой команды ветвления.

В качестве примера рассмотрим программу, вычисляющую и показывающую на экране наибольший общий делитель (НОД) двух натуральных чисел M и N, введенных пользователем. Программа использует алгоритм Евклида, который основывается на том, что НОД пары различных натуральных чисел совпадает с НОД пары, полученной из предыдущей заменой большего числа разностью большего и меньшего чисел.

```
10 REM***НОД***
20 PRINT «Введите M, N»
30 INPUT M, N
40 PRINT «M=»; M; «N=»; N
50 IF M>N THEN M=M-N
60 IF N>M THEN N=N-M
70 IF M<>N THEN GOTO 50
80 PRINT «НОД...»; M
90 END
```

Приведем диалог пользователя с ЭВМ после запуска этой программы.

```
Введите M, N
? 34,51
M=34 N=51
НОД... 17
```

Проследим теперь за работой ЭВМ в ходе ее выполнения. При первом обращении к строке 50 переменная M имеет значение 34, а переменная N — значение 51. Условие M>N не соблюдено, переход к строке 60. Условие N>M соблюдено, N получает новое значение 51—34=17. Условие M<>N (M не равно N) тоже соблюдено и поэтому строка 70 передает управление назад, строке 50.

Теперь M имеет значение 34, а N — значение 17, и условие M>N соблюдено; поэтому переменной M присваивается значение 34—17=17; M=N=17. Условие строки 60 не соблюдено, условие M<>N строки 70 тоже не соблюдено, и ЭВМ наконец выполняет строку 80, выводящую текст «НОД... 17». В строке 90 работа заканчивается.

Эта короткая программа позволяет найти НОД для любой пары натуральных чисел — выполнение серии строк 50, 60, 70 всегда будет повторено необходимое число раз. Об этом «заботятся» включенные в программу команды ветвления.

Рассмотрим теперь, какие условия можно употреблять в команде ветвления. Элементарные условия — это обычные отношения между выражениями:

⟨выражение⟩ ⟨знак отношения⟩ ⟨выражение⟩

В качестве знаков отношения используется < (меньше), > (больше), = (равно),

<= (меньше или равно), >= (больше или равно), <> (не равно). Три последние знака — составные, состоящие из двух символов (на клавиатуре нет специальных клавишей для $\leqslant$, $\geqslant$, $\neq$).

Оба выражения в условии должны быть одного типа: нельзя сравнивать число с символьной строкой! Выполняя команду ветвления, ЭВМ в первую очередь вычисляет значения обоих выражений, потом, сравнивая их, выясняет, соблюдено ли условие. Употребляя в программах числовые условия, особое внимание надо обращать на отношение =, так как может оказаться, что два числа, которые должны быть равными, все-таки немного отличаются (из-за ошибок округления).

Сравнение символьных строк на совпадение (=) или несовпадение (<>) не требует комментариев, остальные их отношения здесь подробно рассматриваться не будут. Грубо говоря, из двух символьных строк меньшей считается та, которая в словаре предшествовала бы другой. Слова на латинском алфавите размещены в таком словаре, как обычно. К сожалению, коды букв русского алфавита не следуют друг за другом по алфавиту (например, не соблюдено условие «ВОВА»<«ГЕНА», так как в ЭВМ буква В следует за буквой Г — в соответствии с латинскими буквами W и G).

В некоторых версиях Бейсика допускаются также составные условия, образованные из элементарных условий, соединенных служебными словами AND (и), OR (или) и NOT (не). Здесь — полное соответствие алгоритмическому языку:

x=0 или x=1 (X=0) OR (X=1)

не (a=2 и b=3) NOT (A=2 AND B=3)

Как уже отмечалось, вся команда ветвления должна разместиться в одной строке программы. Однако не всегда удается втиснуть в одну строку необходимые серии команд. В таких случаях помогает команда GOTO, которая передает управление на строку, где начинается нужный фрагмент программы (если необходимо вернуться к строке, следующей за командой ветвления, то в конце фрагмента опять приходится ставить команду GOTO). Такая конструкция весьма громоздка — поэтому предпочтительнее использовать подпрограммы (команду GOSUB, рассматриваемую в п. 14).

В случае, когда серия состоит лишь из одной команды GOTO, служебное слово GOTO можно опускать, например,

```
100 IF A+B>C*D THEN 150
```

Вопросы.

1. В каких случаях необходима команда ветвления? (Если в зависимости от соблюдения условия необходимо проделать ту или иную операцию.)

2. Когда удобно применять сокращенную форму команды ветвления? (Если серию команд следует выполнять только в случае соблюдения какого-то условия.)

3. Как образуются составные условия? (Из элементарных условий с помощью служебных слов OR, AND, NOT).

Упражнения.

1. Написать команду, по которой печатается $\sqrt{X}$, а если X<0, то выдается соответствующее сообщение.

Ответ.

```
IF X>=0 THEN PRINT SQR (X) ELSE PRINT «x отрицателен»
```

2. Как работает фрагмент программы?

```
100 PRINT «Введите натуральное N»
110 INPUT N
120 IF (N<>INT(N)) OR (N<1) THEN 100
130 PRINT «Спасибо,»; N; «хорошее число»
```

Ответ. Запрашивает ввод числа, отвергает ненатуральные числа. Когда натуральное число — благодарит пользователя.

3. Составить программу, которая соответствует части алгоритма

```
пока x<y
нц
  x:=x+1
кц
Z:=2*x
```

Ответ.

```
100 IF X>=Y THEN Z=2*X : STOP
110 X=X+1: GOTO 100
```

13. Команда повторения с параметром (цикл)

В предыдущем разделе мы убедились, что команда ветвления позволяет организовать циклы — фрагменты программы, выполняющиеся много раз. Теперь, например, программу печати кубов первых 500 натуральных чисел (см. пример в начале п. 11) можно записать в 4 строки:

```
10 I=1
20 PRINT I, I^3
30 I=I+1
40 IF I<=500 THEN 20
```

В этой программе строка 10 устанавливает начальное значение 1 для счетчика i, в строке 30 счетчик получает новое значение, а в строке 40 проверяется, следует ли еще раз повторить выполнение цикла.

Часто даны первый член арифметической прогрессии и разность, и цикл надо выполнить подряд для всех членов прогрессии, не превосходящих заданное конечное значение. В таких случаях удобно использовать коман-

ду повторения с параметром, которая и в Бейсике позволяет записать подобные конструкции короче и нагляднее (программисту не надо специально заботиться об изменении значения счетчика после каждого прохождения цикла).

С ее использованием рассмотренную выше программу можно записать на 3 строках:

```
10  FOR  I=1  TO  500  STEP  1
20  PRINT  I,  I^3
30  NEXT  I
```

Служебные слова — FOR (для), TO (до), STEP (шаг) и NEXT (следующий).

В строке 10 написан «заголовок» цикла: даны начальное и конечное значение счетчика или цикла (если разность прогрессии (шаг) равен 1, то его можно не указывать.) Далее следует «тело» цикла (здесь оно состоит всего из одной строки 20), в котором описана работа, которую нужно проделать для каждого допустимого значения счетчика. Признак конца тела цикла (на алгоритмическом языке это кц) — команда NEXT.

Общий вид заголовка цикла:

FOR ⟨имя переменной⟩ = ⟨выражение1⟩ TO⟨выражение2⟩
[STEP ⟨выражение3⟩]

Часть, заключенную в квадратные скобки, можно опускать,— тогда по умолчанию принимается стандартное значение шага 1. Переменная и все выражения должны быть числовыми.

Опишем работу ЭВМ при выполнении цикла. Сначала ЭВМ отмечает для себя указанное имя счетчика цикла, присваивает в качестве начального значения выражения1, запоминает значение выражения2 (конечное значение счетчика). Если в заголовке присутствует служебное слово STEP, то запоминает значение выражения3 (шаг). Если STEP нет, шаг получает значение, равное 1. Потом ЭВМ выполняет подряд следующие команды, пока не достигнет строки

NEXT ⟨имя переменной⟩

в которой дано запомненное имя счетчика. При выполнении этой команды к значению счетчика добавляется шаг и проверяется, не вышло ли новое значение счетчика из рабочего интервала (если шаг положителен, то не стало ли оно больше конечного значения; если шаг отрицателен — не стало ли меньше конечного значения).

Если счетчик уже перешагнул конечное значение, то работа с циклом закончена и ЭВМ выходит из цикла — забывает имя счетчика, конечное значение и шаг и приступает к выполнению команды, следующей за командой NEXT. Если же счетчик еще не перешагнул конечное значение, то ЭВМ опять начинает выполнять тело цикла, начиная с команды, следующей сразу после заголовка цикла — и все повторяется заново.

Обратите внимание: в любом случае тело цикла выполнится хотя бы один раз!

Рассмотрим примеры.

```
10 FOR X=0 TO 1 STEP .01: PRINT X, SIN (X): NEXT X
```

Эта программа выводит на экран таблицу значений функции для значений аргумента от 0 до 1 с шагом 0.01.

```
10 FOR A=1 TO 10 STEP —1
20  PRINT  A
30  NEXT  A
```

Эта программа отпечатает только начальное значение счетчика A=1. При первой же встрече с NEXT будет обнаружено, что конечное значение уже «позади».

Тело цикла по отношению к остальной части программы замкнуто — не следует «входить» в него, минуя заголовок цикла, и не следует «выходить» из него, минуя NEXT. Из цикла принято выходить только тогда, когда при проверке обнаружено, что счетчик перешел конечное значение. Если надо выйти из цикла раньше, то следует присвоить счетчику значение за границей конечного и командой GOTO выйти на NEXT.

В качестве примера рассмотрим программу, которая должна последовательно просуммировать 20 положительных чисел, введенных пользователем. Как только пользователь введет 0 или отрицательное число, работа прекращается и выводится сообщение ВСЕ!

```
10 REM СУММА 20 ПОЛОЖИТЕЛЬНЫХ ЧИСЕЛ
20  S=0
30  FOR  I=1  TO  20
40  PRINT  «Введите  число»
50  INPUT  K
60  IF  K<=0  THEN  I=21:  S=0: PRINT  «ВСЕ!»  : GOTO  80
70  S=S+K
80  NEXT  I
90  IF  S>0  THEN  PRINT  «Сумма S=»;  S
100  END
```

Приведем пример диалога этой программы с пользователем.

```
RUN
Введите  число
? 10
Введите  число
? 5
Введите  число
? —3
ВСЕ!
Ok
```

Особенно полезна команда цикла при обработке массивов — она позволяет просматривать все элементы массива по порядку.

Рассмотрим программу, которая находит и выводит на экран наибольший и наименьший элементы массива C(0), C(1)..., C(M).

```
10 REM МАССИВ
20  REM===ВВОД  МАССИВА===
30 DIM C(100)
40 PRINT «Введите N<=100»
50 INPUT N
60 FOR I=0 TO N
70 PRINT «C (»; I; «)=»;
80 INPUT C (I)
90 NEXT I
100 REM = = = НАИБОЛЬШИЙ
L, НАИМЕНЬШИЙ M= = =
110 L=C(0): M=C(0)
120 FOR I=0 TO N
130 IF C(I)>L THEN L=C(I)
140  IF  C(I)<M  THEN  M=C(I)
150 NEXT I
160 PRINT «L=»; L, «M=»; M
```


В этой программе оба цикла можно было бы объединить — после ввода элемента сразу же проверить, не является ли он к данному моменту наибольшим или наименьшим — и соответственно изменить значение L или M. Кроме того, если бы не было нужды использовать массив C позже, то его можно было бы и не образовывать — для того, чтобы найти только наибольший и наименьший элементы, незачем запоминать все элементы массива. Однако теперь, когда массив уже в памяти ЭВМ, можно продолжить программу и дополнительно вычислить, например, среднее значение элементом массива:

```
170  REM===СРЕДНЕЕ  ЗНАЧЕНИЕ
===
180 S=0
190 FOR I=0 TO N
200 S=S+C(I)
210 NEXT I
220 PRINT «V=»; S/(N+1)
```

Тело цикла может содержать второй цикл — внутренний по отношению к первому, внешнему циклу. Внутренний цикл должен целиком содержаться в теле внешнего цикла. Тело внутреннего цикла в свою очередь может содержать еще один цикл, внутренний по отношению к нему, и т. д. Интерпретатор Бейсика устанавливает максимальную «глубину» (число) вложения циклов, однако она такая большая, что в простых программах ее не достичь.

В качестве примера рассмотрим программу, которая для данных l, m и n составляет таблицу значений функции

$$f(x)=\sum_{i=1}^{m}\left(\sum_{j=1}^{n}\sqrt{i+jx}\right)^2$$

при x=0, 1, 2, ..., l.

```
10 PRINT «Введите L, M, N»
20 INPUT L, M, N
30 PRINT «M=»; M, «N=»; N
40 FOR X=0 TO L
50 F=0
60 FOR I=1 TO M
70 G=0
80 FOR I=1 TO N
90 G=G+SQR (I+J*X)
100 NEXT J
110 F=F+G*G
120 NEXT I
130 PRINT «F («; X; »)=»; F
140 NEXT X
150 END
```

Вопросы.

1. В каких случаях используется команда цикла? (Если надо повторить действия для всех членов заданной арифметической прогрессии.)

2. Что такое счетчик цикла? (Это переменная, имя которой указано в заголовке цикла; выполнение тела цикла повторяется для всех значений счетчика от начального до конечного.)

3. Что произойдет, если программист не укажет величину шага?
(Автоматически будет установлен шаг 1.)

Упражнения.

1. Написать программу, которая создает массив M(0)=1000, M(1)=999,..., M (1000)==0.

Ответ.

```
10 DIM M(1000)
20 FOR I=0 TO 1000
30      M(I)=1000—I
40 NEXT I
```

2. Что напечатает программа?

```
10 A=0
20 FOR L=100 TO 15 STEP —10
30 A=A+L
40 NEXT L
50 PRINT A
```

Ответ: число 540

3. Что напечатает программа?

```
10 DIM S$(32)
20 FOR K1=0 TO 30 STEP 3
30 S$(K1)=«A»
40 S$ (K1+1)=«B»
50 S$(K1+2)=«C»
60 NEXT K1
70 FOR K2=0 TO 32
80 PRINT S$(K2);
90 NEXT K2
```

Ответ. Строку ABCABCABCABCABC ABCABCABCABCABCABC

14. Подпрограммы

При составлении сложных программ удобно использовать подпрограммы (в алгоритмическом языке им соответствуют вспомогательные алгоритмы). Это позволяет при необходимости «вызвать» нужный фрагмент программы (подпрограмму) много раз, в том числе и из разных мест программы. После того, как подпрограмма выполнена, управление передается обратно на место, откуда она была вызвана, и работа по основной программе продолжается.

Переход на подпрограмму осуществляется по команде

GOSUB ⟨номер строки⟩

(GO to SUBroutine — идти к подпрограмме). Выполняя эту команду, ЭВМ запоминает «адрес возврата» — номер строки, следующей за данной командой GOSUB и приступает к выполнению команд, записанных со строки с указанным в GOSUB номером. Только встретив команду RETURN (RETURN — возвращаться), ЭВМ возвращается назад, к запомненному адресу. Таким образом, команда RETURN служит признаком конца подпрограммы.

Команда GOSUB может показаться очень похожей на команду GOTO. Существенная разница между ними в том, что GOSUB как бы «посылает в командировку» — после выполнения задачи происходит возвращение назад.

Если GOTO делает программу непонятной, то GOSUB, наоборот, позволяет записать «план» алгоритма коротко, а подробности перенести в подпрограммы, делая основную программу наглядной и обзорной.

В качестве примера рассмотрим составление программы, которая запрашивает и проверяет на натуральность два числа, вычисляет их наименьшее общее кратное (НОК), причем позволяет пользователю повторить нахождение НОК. Задача естественно разбивается на этапы: сначала надо ввести числа, потом вычислить НОК. Запишем этот план в качестве основной программы.

```
10 REM НОК — Наименьшее Общее Кратное
20 REM IKAUNIEKS 1 — MAR — 86
30 PRINT «Вычисление НОК»
40 GOSUB 1000:REM-Ввод чисел
50 GOSUB 2000:REM=Вычисление НОК=
60 PRINT «Повторим?» (да/нет)»
70 INPUT B$
80 IF B$=«да» THEN 30
90 STOP
```

Структура этой программы настолько проста, что можно сразу перейти к написанию подпрограмм. В более сложных случаях желательно сначала отладить основную программу и только потом браться за подпрограммы; для этого можно вместо каждой подпрограммы вставить «пробку» — несколько команд, которые только имитируют выполнение подпрограммы. Так, например, вместо подпрограммы ввода чисел можно вставить

```
1000 REM SUB — Ввод чисел
1010 M=12: N=18
1020 RETURN
```

Напишем теперь настоящие подпрограммы.

```
1000 REM SUB — Ввод чисел
1010 GOSUB 3000: REM=Получение X=
1020 M=X
1030 GOSUB 3000:REM=Получение X=
1040 N=X
1050 PRINT «M=»; M, «N=»; N
1060 RETURN
```

В этой подпрограмме мы вызываем еще одну подпрограмму, так как нам надо два раза повторить одну и ту же работу (запросить и проверить число). Сразу же составим ее:

```
3000 REM SUB — Получение X
3010 PRINT «Введите натуральное X»
3020 INPUT X
3030 IF X<>INT (X) THEN PRINT «Целое!»; GOTO 3020
3040 IF X<=0 THEN PRINT «Положительное!»: GOTO 3020
3050 RETURN
```

Осталось написать подпрограмму вычисления НОК. В алгоритме используем то, что произведение НОК и НОД равно произведению самих чисел (переменная K).

```
2000 REM SUB — Вычисление НОК
2010 K=M*N
2020 IF M>N THEN M=M—N
2030 IF N>M THEN N=N—M
2040 IF M<>N THEN 2020
2050 PRINT «НОК....»; K/M
2060 PRINT
2070 RETURN
```

Собрав вместе все эти подпрограммы, получаем окончательный текст нашей программы. Для того чтобы сделать листинг программы более наглядным, можно отделять тексты подпрограмм друг от друга фиктивным комментарием

```
100 REM— — — — — — — — —
1070 REM — — — — — — — — —
2080 REM — — — — — — — — —
```

Как мы уже видели, подпрограмма может, в свою очередь, вызвать другую подпрограмму, та — другую и т. д. Наглядно это можно представить себе следующим образом. Пусть меня из Риги послали в Москву на повышение квалификации, а оттуда еще в Ленинград на курсы. По окончании этих кур-

сов я возвращаюсь в Москву, заканчиваю учебу там и только после этого возвращаюсь домой.

Вопросы.

1. Для каких целей необходимы команды GOSUB и RETURN?
(Для вызова вспомогательных алгоритмов-подпрограмм.)

2. Чем отличаются команды GOSUB и GOTO? (При выполнении GOSUB происходит автоматическое возвращение назад, при GOTO — нет.)

Упражнения.

1. Проследите историю работы по программе вычисления НОК на примере M=12, N=18.

Ответ. Последовательно выполняются следующие строки: 10,20,30,40, (1000,1010, (3000, 3010,3020,3030,3040,3050), 1020,1030, (3000,3010, 3020, 3030, 3040, 3050), 1040, 1050, 1060), 50, (2000, 2010, 2020, 2030, 2040, 2020, 2030, 2040, 2050, 2060, 2070), 60,70, 80,90.

2. Что напечатает программа?

```
10 REM — — — ГОРОДА — — —
20 PRINT «Живу в Риге»
30 GOSUB 1000
40 PRINT «Я снова дома»
50 END
60 REM
1000 REM — SUB — МОСКВА —
1010 PRINT «Приехал в Москву»
1020 GOSUB 2000
1030 PRINT «Возвратился в Москву»
1040 RETURN
1050 REM
2000 REM — SUB — ЛЕНИНГРАД —
2010 PRINT «Приехал в Ленинград»
2020 RETURN
```

Ответ.

```
Живу в Риге
Приехал в Москву
Возвратился в Москву
Я снова дома
```

15. Стандартные функции программиста

Вспомогательным алгоритмам вычисления значений функций в алгоритмическом языке в Бейсике соответствуют стандартные функции, определяемые программистом. В большинстве версий, однако, можно определить только функции, значение которых задается выражением (на алгоритмическом языке допускаются произвольно сложные вспомогательные алгоритмы).

Итак, если в программе необходимо многократно вычислять значение функции, которая не является стандартной функцией Бейсика, то программист может сам определить ее и потом использовать в своей программе наравне со стандартными функциями Бейсика.

Имя функции программиста должно состоять из трех букв латинского алфавита, первые две из них — FN. Таким образом, можно определить 26 числовых функций с именами FNA,... FNZ и 26 литерных функций с именами FNA$, ..., FNZ$ (в различных версиях Бейсика здесь могут быть варианты). Функцию можно определить командой DEF (DEFine — определить). Общий вид команды DEF ⟨имя функции⟩ (⟨аргументы⟩)=⟨выражение⟩

Примеры:

```
DEF FNC (Z)=5 * Z * Z — 4* Z+3
DEF FNK (X)=A * EXP(X) — B * SIN
(2*X)
DEF FNM$(X$)=MID$(X$,2,LEN(X$)—
—1)
DEF FNS(X,Y)=LOG(X*X+Y*Y)
```

В ходе выполнения программы команды DEF пропускаются, однако каждый раз, когда понадобится значение функции программиста, это значение будет вычислено согласно определению функции в соответствующей команде DEF.

В качестве примера рассмотрим фрагмент программы, который для стандартной функции программиста находит корень на данном интервале. Используется метод деления интервала пополам. В строках 510 и 520 можно задать концы интервала (в них значения функции должны иметь разные знаки), а в строке 530 — требуемую точность.

```
500 DEF FNF(X)=COS(X)—2 * X
510 A=0
520 B=2
530 E=.0001
540 IF SGN (FNF(A)) * SGN(FNF(B))>
>0 THEN PRINT «Знаки!»:
550 PRINT «Корень...»;
560 C=(A+B)/2
570 IF FNF (A)=0 THEN PRINT A:END
580 IF FNF(B)=0 THEN PRINT B:END
590 IF ABS(B—A)<E THEN PRINT C
:END
600 IF SGN (FNF(A))<>SGN(FNF(C))
THEN B=C:GOTO 560
610 A=C:GOTO 560
```

Вопросы:

1. В каких случаях программисту выгодно определять новую функцию? (Если часто приходится вычислять значение одного выражения.)

2. Как быть с переменными, имена которых встречаются в определяющем функцию выражении, но не упомянуты в списке аргументов? (при вычислении значения функции они заменяются на текущие значения соответственных переменных.)

Упражнения.

1. Напишите фрагмент программы, в котором определяется функция $g(x)=\sqrt{|x^2-1|}$ и печатается таблица значений этой функции при x=0, .1, .2, ...2.

Ответ.

```
100 DEF FNG(X)=SQR(ABS(X*X-1))
110 FOR X=0 TO 2 STEP .1
120 PRINT X, FNG (X)
130 NEXT X
```

2. Каков результат работы программы?

```
10 A=3
20 DEF FNA(X)=X * X-A
30 DEF FNB (Y)=5+Y
40 PRINT FNB (FNA(4))
```

Ответ. Будет отпечатано число 18.

16. Рекомендации по переводу с алгоритмического языка на Бейсик

Общий вид алгоритма. В Бейсике нет таких строгих правил оформления программы, как в алгоритмическом языке. Желательно в первых строках программы в комментариях дать короткое объяснение сущности алгоритма и используемых переменных.

Общий вид алгоритма вычисления функций. В Бейсике программист может определить только простейшие функции, задаваемые выражением (см. п. 15). В более сложных случаях приходится пользоваться командой GOSUB (п. 14).

Пусть требуется вычислить c=ln((m—1)!). Используя алгоритм вычисления факториала (см. алгоритм 5 из Приложения к пробному учебному пособию), можно на алгоритмическом языке записать команду:

c:=ln (факториал (m—1))

На Бейсике соответствующий фрагмент программы будет использовать подпрограмму вычисления факториала (допустим, что она расположена со строки 500):

```
N %=M %-1:GOSUB  500:C=LOG(Z %)
```

Команды ветвления.

IF⟨условие⟩ THEN ⟨серия1⟩ ELSE (серия2⟩

IF ⟨условие⟩ THEN ⟨серия⟩

Команды ветвления переносятся без изменений. Трудности могут возникнуть лишь в связи с тем, что вся команда должна размещаться в одной логической строке (см. п. 10).

Команда выбора. Сокращенную форму можно просто реализовать отдельными командами ветвления:

IF ⟨условие1⟩ THEN ⟨серия1⟩

IF ⟨условие 2⟩ THEN ⟨серия 2⟩

IF ⟨условие n⟩ THEN ⟨серия n⟩

Если в полной форме случай несоблюдения ни одного из условий 1, ..., n удается просто выразить одним условием (условие иначе), то в Бейсике можно добавить к предыдущему

IF ⟨условие иначе⟩ THEN ⟨серия⟩

Однако такой вариант может дать ошибку, если в сериях после THEN меняются значения переменных, входящих в условия. Этого можно избежать, используя «флажок» F:

```
F=0
IF ⟨условие1⟩ THEN F=1: ⟨серия1⟩
IF ⟨условие2⟩ AND F=0 THEN F=1:
:⟨серия2⟩
- - - - - - - - - - - - - - - - -
IF ⟨условие n⟩ AND F=0 THEN F=1:
:⟨серия n⟩
IF F=0 THEN ⟨серия⟩
```

Команда повторения.

⟨№ строки⟩ IF ⟨условие⟩ THEN ⟨серия⟩: :GOTO ⟨№ строки⟩

Пример:

```
100 IF X>=0 THEN X=X-10:GOTO 100
```

Если серия не умещается в данной строке, можно серию перенести на подпрограмму, вызываемую командой GOSUB:

```
500 IF A>B THEN GOSUB 1000:GOTO 500
```

Команда, повторения с параметром.

```
FOR  X=⟨начало⟩  TO  ⟨конец⟩STER
⟨шаг⟩ ⟨серия⟩
NEXT X
```

Команда присваивания.

⟨имя переменной⟩=⟨выражение⟩

Команда вызова вспомогательного алгоритма. В большинстве версий Бейсика вызов подпрограмм организуется только командой

GOSUB ⟨номер строки⟩

О передаче значений параметров подпрограмме и передаче результатов работы подпрограммы главной программе должен заботиться сам программист.

Пример.

```
- - - - - - - - - - - - - -
100 A=1:B=2
110 GOSUB 1000
120 PRINT S
- - - - - - - - - - - - -
500 END
- - - - - - - - - - - - -
1000 REM По данным A и B вычисляет S

1100 RETURN
- - - - - - - - - - - - -
```

Запись условий. Трудности при переводе условий возникают только в версиях, где нет составных условий. Но и тогда соблюдение составных условий можно моделировать с помощью флажков:

```
IF ⟨усл1⟩ AND ⟨усл2⟩ THEN ⟨серия⟩
можно заменить на
F=0:G=0
IF ⟨(усл1⟩ THEN F=1
IF ⟨усл2⟩ THEN G=1
IF F * G>0 THEN ⟨серия⟩
```

Условие ⟨усл1⟩ OR ⟨усл2⟩ моделируется

аналогично, только вместо F * G>0 надо брать F+G>0.

17. Задачи

1. Написать программу, которая приглашает ввести два целых числа X и Y, вычисляет (не употребляя операцию X^ Y) и печатает X в степени Y.

```
10 REM — PR 1 — ВОЗВЕДЕНИЕ
                    В СТЕПЕНЬ —
20 PRINT «Введите целые X, Y»
30 INPUT X, Y
40 IF X=0 THEN IF Y>0 THEN PRINT 0
        ELSE PRINT «Нельзя!»: STOP
50 IF Y=0 THEN PRINT 1:STOP
60 IF Y<0 THEN Y=—Y: X=1/X
70 Z=1
80 FOR I=1 TO Y
90 Z=Z*X
100 NEXT I
110 PRINT Z
120 END
```


2. Написать программу, проверяющую знание пользователем таблицы умножения — задаются вопросы вида «m*n=?» (m и n — случайные цифры), проверяются и оцениваются ответы.

```
10 REM — PR2 — УМНОЖЕНИЕ —
20 PRINT «Введите свое счастливое
                        число»
30 INPUT X
40 X=RND (ABS(X))
50 FOR I=1 TO 5
60 X=INT (RND(1) * 9)+1
70 Y=INT (RND(1)*9)+1
80 PRINT X; «*»; y; «=?»
90 INPUT Z
100 IF Z=X*Y THEN PRINT «Да!»
        ELSE PRINT «Нет...»
110 NEXT I
120 END
```

3. Треугольники Пифагора. Написать программу, которая для введенного пользователем натурального числа с находит и печатает все пары натуральных чисел (A, B) такие, что $A^2+B^2=C$ (если такие существуют).

```
10 REM — PR3 — ТРЕУГОЛЬНИКИ
                    ПИФАГОРА —
20 PRINT «Гипотенуза C»
30 INPUT C
40 IF C <> INT (C) OR C<2 THEN 20
50 PRINT «C=»; C
60 F=0
70 FOR A=1 TO SQR(C * C/2)
80 B=SQR(C * C—A * A)
90 IF B=INT(B) THEN
        PRINT «Катеты»; A; B:F=1
100 NEXT A
110 IF F=0 THEN PRINT «Нет решений»
120 END
```

4. Написать программу, которая задает пользователю вопросы вида «Два угла треугольника — m и n градусов, чему равен третий?» (m и n — случайные числа), проверяет и оценивает ответы.

```
10 REM — PR4 — УГЛЫ
                    ТРЕУГОЛЬНИКА —
20 X=RND(—TIME):REM = Разные
                случайные числа=
30 M=INT(RND(1)*178+1)
40 N=INT(RND(1)*(179—M)+1)
50 PRINT «Углы в»; M «и»; N;
                        «градусов»
60 PRINT «Введите третий угол
                    треугольника»
70 INPUT L
80 IF L<>180—M—N THEN
        PRINT «Нет...»:GOTO 50
90 PRINT «Хорошо. Продолжим?
                        (да/нет)»
100 INPUT A$.
110 IF A$=«да» THEN 30
120 END
```

5. Написать программу, которая по введенным пользователем числам a, в, c выясняет, существует ли треугольник со сторонами a, в, c, и если существует, вычисляет его площадь.

```
10 REM — PR5 — ПЛОЩАДЬ
                    ТРЕУГОЛЬНИКА —
20 PRINT «Введите стороны
                треугольника A, B, C»
30 INPUT A, B, C
40 IF A>=B+C OR B>=C+A OR C>=
                        =A+B
                    THEN
        PRINT «Нет»: STOP
50 P=(A+B+C)/2
60 PRINT «Площадь»;
    SQR (P*(P—A)*(P—B)*(P=C))
70 END
```

6. Написать программу, которая приглашает пользователя ввести 10 чисел, выясняет, образуют ли они арифметическую прогрессию и печатает соответствующее сообщение.

```
10 REM — PR6 — АРИФМЕТИЧЕСКАЯ
                    ПРОГРЕССИЯ —
20 DIM P(10)
30 FOR I=1 TO 10
40 PRINT «P(»; I; »)=?»
50 INPUT P(I)
60 NEXT I
70 F=1
80 FOR I=2 TO 9
90 IF P(I)<>(P(I—1)+P(I+1))/2
                    THEN F=0
100 NEXT I
110 IF F=0 THEN PRINT «НЕ»
120 PRINT «Арифметическая прогрессия»
130 END
```

7. Написать программу, которая для введенной пользователем суммы печатает способ выдачи этой суммы наименьшим числом монет (имеются монеты 1, 2, 3, 5, 10, 15 и 20 коп.).

```
10 REM — PR7 — МОНЕТЫ —
20 DATA 20, 15, 10, 5, 3, 2, 1
30 PRINT «Введите сумму в копейках»
40 INPUT S
50 IF S<>INT (S) OR S<1 THEN 20
60 PRINT «Разменяю —»
70 FOR I=1 TO 7
80 READ M
90 K=INT (S/M)
100 IF K>0 THEN
                PRINT K; «раз»; M; «коп.»
110 S=S—K * M
120 NEXT I
130 END
```

8. Написать программу, которая вычисляется и печатает первые n+1 членов последовательности Фибоначчи a_0, a_1, a_2, ..., a_n ($a_0=1$, $a_1=1$, $a_{k+1}=a_k+a_{k-1}$). Значение n вводит пользователь.

```
10 REM — PR8 — «Последовательность Фибоначчи» —
20 PRINT «Введите N»
30 INPUT N
40 IF N<>INT(N) OR N<2 THEN 20
50 I=2: A=1: B=1
60 PRINT O, A
70 PRINT 1,B
80 PRINT I, A+B
90 I=I+1: B=A+B: A=B—A
100 IF I<=N THEN 80
110 END
```

9. Написать программу, которая печатает разложение введенного пользователем натурального числа на простые сомножители, например, 24=2 * 2 * 2 * 3 * , 17=17

```
10 REM — PR9 — ПРОСТЫЕ
                СОМНОЖИТЕЛИ —
20 PRINT «Введите натуральное N, N>1»
  30 INPUT N
  40 PRINT N; «=»;
  50 P=2: S$=««+»»
  60 D=N/P
  70 IF D=INT (D) THEN
    PRINT S$; P;:S$=«*»:N=D:GOTO 60
  80 IF D<>INT(D) THEN P=P+1
  90 IF N>=P THEN 60
  100 PRINT
  110 END
```

10. Последовательность a_0, a_1, a_2... образуется по закону

$$a_{n+1}=\begin{cases} a_n/2, & \text{если } a_n \text{ — четное число} \\ 3a_n+1, & \text{если } a_n \text{ — нечетное число.} \end{cases}$$

Написать программу, которая для введенного пользователем натурального a_0 печатает последовательность до первого ее члена, равного 1.

```
10 REM — PR10 — ПОСЛЕДОВАТЕЛЬНОСТЬ ДО 1 —
  20 PRINT «Введите натуральное A»
  30 INPUT A
  40 PRINT A
  50 IF A=1 THEN STOP
  60 IF A/2=INT (A/2)
          THEN A=A/2 ELSE A=3*A+1
  70 GOTO 40
```

11. Написать программу, которая, задав пользователю 4 вопроса (ответы только «да» или «нет»), угадает его месяц рождения.

12. Шарик, подброшенный вверх с начальной скоростью V_0, достигает высоты $H=V_0^2/2G$. Написать программу игры, которая случайно выбирает желаемую высоту, сообщает ее пользователю, запрашивает ввод начальной скорости, а потом сообщает результат: низко, высоко или хорошо (с требуемой точностью).

13. Написать программу, которая с указанием цен предлагает пользователю покупать хлеб, молоко, сыр и т. д., запрашивает ввод количества покупаемых товаров и выдает суммарную стоимость покупок.

14. Написать программу, которая по введенному пользователем порядковому номеру дня в году печатает дату этого дня.

15. Написать программу, которая для введенной пользователем даты (в рамках XX века — по новому стилю) вычисляет день недели. Форма печати результатов —

«3 марта 1986 года — понедельник».

16. Написать программу, которая в подготовленный заранее шаблон письма вписывает введенные пользователем дату, адрес, подпись и прочее, а потом печатает готовое письмо.

17. Написать программу, которая находит, между какими двумя последовательными степенями двойки лежит введенное пользователем натуральное число. Примеры печати: 2^4<24<2^5, 128=2^7.

18. Особенности Бейсика на ДВК-2М и СМ-4

Приведем отличия версии Бейсика на ДВК-2М и СМ-4 от версии MSX, рассмотренной выше.

На клавиатуре этих ЭВМ отсутствуют строчные буквы. Некоторые знаки имеют другую графическую форму: знак возведения в степень выглядит как — (а не ^), валютный знак, используемый как признак имени литерной переменной, выглядит как ¤ (а не $).

Литерные строки можно заключать в кавычки, а можно и в апострофы. Так, например, 'ЭВМ «Агат»' — это литерная строка, состоящая из 10 символов (в том чис-

ле двух кавычек), а «Don't cry» — литерная строка из 9 символов (содержащая апостроф).

Если в MSX Бейсике можно использовать и длинные имена переменных, то здесь допускаются только имена, описанные в п. 3.

В качестве знака соединения литерных строк, кроме +, можно использовать и &.

В качестве разделителя нескольких команд, расположенных в одной строке, используется обратная косая черта \ (а не:), например,

```
10 A=1 \ B=2 \ C=3
```

Нет стандартной функции TAN (X), приходится в выражениях вместо tg x записывать SIN(X)/COS(X). Зато имеется стандартная постоянная PI, имеющая значение π. Ее можно свободно использовать в выражениях, например, SIN (PI/3).

Имеется стандартная функция LOG10 (X) для десятичного логарифма lgx.

В стандартной функции RND аргумент не влияет на значение функции, аргумент можно даже опускать, например, X=RND. Если нужно, чтобы при запуске программы каждый раз получалась другая последовательность случайных чисел, можно использовать команду RANDOMIZE. Например,

```
100 RANDOMIZE
110 N=INT (10*RND)
```

Нет стандартных функций LEFT $, RIGHT$ и MID$. Роль MID$ играет команда

SEG$(X$, M, N),

в которой N уже не длина вырезки, а номер последнего символа, например,

SEG$ («КИБЕРНЕТИКА», 3, 6) = «БЕРН».

Разумеется, в именах литерных функций $ должно быть заменено на $.

В командах записи и чтения программ имена программ не заключаются в кавычки. Примеры записи во внешнюю память:

```
SAVE PROG 1
SAVE NAME
```

При загрузке из внешней памяти в оперативную используется команда OLD (старый), а на LOAD, например,

```
OLD PROG1
OLD NAME
```

19. Список служебных слов Бейсика

Здесь приведены служебные слова, рассмотренные в статье, их рекомендуемое произношение, значения и номера разделов, в которых они объясняются или используются.

ABS(x)	абс	Абсолютное значение	4
AND	энд	И	12
ASC(x$)	аск	Код символа X$	5
ATN(X)	атэн	Арктангенс	4
CHR$(X)	цехаэр	Символ с кодом X	5
COS(X)	кос	Косинус	4
DIM	дим	Размерность массива	3
ELSE	элс	Иначе	11
AND	энд	Конец	11
EXP(X)	эксп	Показательная функция	4
FOR	фор	Для	13
GOSUB	гоусаб	Идти к подпрограмме	15
GOTO	гоуту	Идти к	11
IF	иф	Если	12
INPUT	инпут	Ввод информации	7
INT(X)	инт	Целая часть	4
LEFT$(X$,M)	лефт	Начало строки	5
LEN(X$)	лен	Длина текста	5
LET	лэт	Присваивание	6
LIST	лист	Перелистать	10
LOAD	лоуд	Загрузить	10
LOG(X)	лог	Натуральный логарифм	4
MID$(X$,M,N)	мид	Вырезка из текста	5
NEW	нью	Новый	10
NEXT	некст	Следующий	13
NOT	нот	Не	12
OR	ор	Или	12
PRINT	принт	Вывод информации	7
READ	рид	Читать	14
REM	рем	Заметка	10

RETURN	ритён	Возвращаться	15
RIGHT$ (X$,M)	райт	Конец строки	5
RND	эрэндэ	Случайное число	4
RUN	ран	Выполнить	8
SAVE	сейв	Сохранить	10
SIN(X)	син	Синус	4
SGN(X)	эсгээн	Знак	4
SQR(X)	экскьюэр	Квадратный корень	4
STEP	стэп	Шаг	13
STOP	стоп	Остановиться	11
STR$(X)	эстээр	X как строка	5
TAN(X)	тан	Тангенс	4
THEN	зэн	То	12
TO	ту	До	13
VAL(X$)	вэл	Значение	5

Л. ВИКЕНТЬЕВ, О. КОЗЛОВ

Алгоритмический язык на ЕС ЭВМ

Алгоритмический язык, изучаемый в школьном курсе, позволяет наглядно представлять алгоритмы с использованием русских ключевых слов, что ставит его практически вне конкуренции в качестве языка публикаций, служащего для обмена алгоритмами между людьми.

В то же время авторы алгоритмического языка не планировали его «машинное» использование, т. е. не рассматривали как средство, предназначенное для обмена алгоритмами между людьми и ЭВМ. Это выразилось, в частности, и в том, что в языке не формализована такая фундаментальная конструкция, как «выражение», не предусмотрены какие-либо средства для организации обмена информацией между человеком и вычислительной машиной в процессе выполнения программы (алгоритма).

Школьный курс во многом ориентирован на использование персональных вычислительных средств, однако до оснащения всех школ такой техникой пока еще далеко, поэтому естественным выглядит появление работ по адаптации школьного алгоритмического языка к наиболее распространенной в нашей стране вычислительной технике. В первую очередь следует указать на создание Е-практикума.

В статье описывается первая версия разработанного авторами алгоритмического языка персональных вычислений (ПВ). Он ориентирован на непрограммирующих профессионалов (инженеров, врачей, юристов, учителей и т. п.) и предназначен для формализации их профессиональных знаний, для решения ими своих профессиональных задач с помощью вычислительных машин. Вместе с тем, поскольку язык ПВ основан на применении тех же ключевых слов, что и школьный алгоритмический язык, его программное обеспечение в полной мере может быть использовано в процессе изучения курса «Основы информатики и вычислительной техники».

Средства программной поддержки языка персональных вычислений реализованы на ПЛ/1 применительно к ОС ЕС ЭВМ, что позволяет уже сейчас использовать мощный парк существующей вычислительной техники серии ЕС для помощи школе, тем более что ЕС ЭВМ в целом по стране загружены недостаточно.

Краткое описание языка ПВ

Алфавит языка ПВ включает в себя цифры, буквы (прописные латинского

и русского алфавитов) и специальные символы: ␣, +, —, *, /, =, (,), [,], ., ′, :, ?, %, >, <, !, &, ¬, запятая.

В отличие от алгоритмического языка школьного курса ключевые слова в языке ПВ выделяются не подчеркиванием, а точками слева и справа от ключевого слова, например: .АЛГ., .НАЧ., .ЕСЛИ., .ПОКА., .КОН. и т. п. Описываемая версия (версия 1.1.) позволяет использовать только те ключевые слова и конструкции, которые использованы в пробном учебном пособии для IX класса по курсу «Основы информатики и вычислительной техники». Другим ее ограничением является запрещение использования букв, относящихся только к русскому алфавиту (Б, Г, Д, Ж, З, И, Л, П, У, Ф, Ц, Ч, Щ, Ш, Ь, Ъ, Ы, Э, Ю, Я) в составе имен объектов программы. Таким образом, для рассматриваемой версии языка ПВ в качестве имен объектов программы (идентификаторов) разрешается использовать лишь последовательность символов, первым из которых является прописная буква латинского алфавита, а последующим — прописные буквы латинского алфавита или цифры. Например, M5, A, X2B15, RADIUS, BLOK15, T328 и т. п.

Длина идентификаторов, являющихся именами внутренних объектов, не должна превышать 31 символа, а имени всей ПВ-программы — 7 символов.

Данные. Под данными ПВ-программы понимаются некоторые постоянные и переменные величины, подлежащие обработке при ее выполнении.

По своему назначению данные делятся на арифметические, символьные, логические и управляющие. Арифметическими называются данные, значениями которых являются числа. Символьными называются данные, значениями которых являются строки символов, заключенные в апострофы, например: 'КОНЕЦ РЕШЕНИЯ', 'ОБ', 'ЕМ', 'СКОРОСТЬ', 'ALPHA', 'БЛОК-18' и т. п. Логическими называются данные, принимающие только два значения: истина (1) или ложь (0). Управляющие данные применяются для управления выполнением программы и в настоящей статье не рассматриваются.

Арифметические данные в свою очередь подразделяются на точные (представляемые в ЭВМ абсолютно точно) и приближенные (представляемые в ЭВМ с некоторой ограниченной точностью). К первым относятся данные, имеющие тип .ЦЕЛ. или .НАТ., ко вторым — имеющие тип .ВЕЩ..

Все данные подразделяются на константы и переменные. Обозначаемые непосредственно своими значениями и не изменяющиеся в процессе выполнения программы данные называют константами, обозначаемые своими именами (идентификаторами) и могущие изменяться в процессе выполнения программы — переменными.

Арифметические константы, имеющие тип .ЦЕЛ. или .НАТ., представляются в виде некоторой последовательности цифр (перед целыми числами может быть записан знак + или —), незначащие нули могут быть опущены, например: 0, 25, —14, +170, 1986, 01, 1 и т. п. Арифметические константы, имеющие тип .ВЕЩ., представляются либо в форме с фиксированной точкой, либо в форме с плавающей точкой. В первом случае константа имеет целую и дробную части, разделенные точкой, во втором — справа к числу (называемому мантиссой) приписывается буква E, после которой записывается порядок — целое число, указывающее, на какую степень числа 10 следует умножить мантиссу, чтобы получить требуемую константу; если мантисса не содержит точки, то подразумевается, что точка находится после последней цифры. Например, константы 2.5, —0.37, .15, —1,15 E2, +1.25E—2 и 3E+10 обозначают соответственно вещественные числа: +2,5; —0,37; +0,15; —115,0; +0,0125 и +30 000 000 000,0.

Операции. При преобразовании данных с ними могут выполняться операции взятия элемента таблицы, арифметические, сравнения, логические и соединения строк (см. табл. 1).

Отметим, что обозначения некоторых операций, например возведения в степень (* *), соединения строк (!! , сравнения на неравенство (.НЕ.= или ¬=) и т. д., являются составными. В таких операциях составляющие символы нельзя разделить пробелами,

Таблица 1

Название операции	Обозначение	Примечания
1. Операции взятия элемента таблицы		X — имя (идентификатор) таблицы I, K — номера (индексы) элемента
1.1. Взятие элемента линейной таблицы	X[I]	
1.2. Взятие элемента прямоугольной таблицы	X[I,K]	
1.3. Взятие элемента многомерной таблицы	X[I, J, .., K]	
2. Арифметические операции		A, B — арифметические выражения
2.1. Сложение	A+B	
2.2. Вычитание	A—B	
2.3. Умножение	A*B	
2.4. Деление	A/B	
2.5. Возведение в степень	A**B	
3. Операции сравнения		A, B — арифметические выражения
3.1. Сравнение на равенство	A=B	
3.2. Сравнение на неравенство	A.НЕ.=B или A¬=B	
3.3. Сравнение на «больше»	A>B	
3.4. Сравнение на «меньше»	A<B	
3.5. Сравнение на «не больше»	A.НЕ.>B или A¬>B	
3.6. Сравнение на «не меньше»	A.НЕ.<B или A¬<B	
3.7. Сравнение на «больше или равно»	A>=B	
3.8. Сравнение на «меньше или равно»	A<=B	
4. Логические операции		A, B — логические выражения
4.1. Отрицание	НЕ.A или ¬A	
4.2. Конъюнкция, или логическое умножение	A.И.B или A&B	
4.3. Дизъюнкция, или логическое сложение	A.ИЛИ.B или A!B	
5. Операция соединения строк	A!!B	A, B — символьные (литерные) строки

нельзя записывать их на разных строках, их можно писать только непосредственно друг за другом в одной строке.

Стандартные функции. При преобразовании данных в ПВ-программах допускается непосредственное использование любых стандартных функций базового алгоритмического языка ПЛ/1.

Эти функции вызываются путем записи в выражении специальной конструкции — указателя функции, имеющего структуру F(A) или F(A, B, ..., C), где F — имя (идентификатор) функции; A, B, ..., C — выражения, задающие значения аргументов. Основные стандартные функции языка ПВ приводятся в табл. 2.

Выражения арифметического, логического или символьного типов в языке ПВ строятся по следующим индуктивным правилам.

Константа, указатель функции, идентификатор переменной или элемент таблицы любого типа являются выражениями того же типа.

Два арифметических выражения, соединенных знаком операции сравнения, являются логическим выражением.

Выражение любого типа, будучи заключенным в скобки, остается выражением того же типа.

Арифметическое выражение, перед которым ставится знак + или —, логическое выражение, перед которым ставится знак .НЕ., остаются выражениями того же типа.

Два арифметических выражения, соединенных знаком арифметической операции, два логических выражения, соединенных знаками логических операций .И. или .ИЛИ., два символьных выражения, соединенных знаком опера-

Таблица 2

Математическая запись функции	Запись указателя функции в языке ПВ	Примечания
$\sqrt{A}$	SQRT(A)	A⩾0
e^A	EXP(A)	
lg A=$\log_{10}$A	LOG10(A)	A>0
ln A=$\log_e$A	LOG(A)	A>0
ldA=$\log_2$A	LOG2(A)	A>0
sin A	SIN(A)	A — в радианах
	SIND(A)	A — в градусах и долях градуса
sh A	SINH(A)	
cos A	COS(A)	A — в радианах
	COSD(A)	A — в градусах и долях градуса
ch A	COSH(A)	
tg A	TAN(A)	A — в радианах
	TAND(A)	A — в долях градуса и градусах
th A	TANH(A)	
arctg A	ATAN(A)	Значение arctg A в радианах
	ATAND(A)	Значение arctg A в градусах и долях градуса
|A|	ABS(A)	
sign A	SIGN(A)	
max(A, B, ..., C)	MAX(A, B, ..., C)	Число аргументов от 2 до 64
min(A, B, ..., C)	MIN(A, B, ..., C)	То же
Вырезка части строки	SUBSTR(A, I, K)	Вырезает из символьной строки A K символов, начиная с 1-го

ции соединения строк, являются выражениями того же типа.

Других выражений в языке ПВ не существует.

Последовательность выполнения операций при вычислении значения некоторого выражения определяется следующими правилами.

Операции, записанные внутри скобок, выполняются в первую очередь (операции в квадратных скобках — вычисление индексов элементов таблиц — выполняются раньше, чем операции в круглых скобках).

В первую очередь вычисляются значения функциии по их указателям.

Выполняются одноместные операции присвоения знака (+ и —) и отрицания (.НЕ.), а также операция возведения в степень (выражение просматривается от конца к началу).

Выполняются в порядке их записи операции умножения и деления.

Выполняются в порядке их записи операции сложения и вычитания.

Выполняются в порядке их записи операции соединения строк.

Выполняются в порядке их записи операции сравнения.

Выполняются в порядке их записи операции логического умножения (.И.).

Выполняются в порядке их записи операции логического сложения (.ИЛИ.).

В качестве иллюстрации рассмотрим вычисление значения выражения

—A/B+0.25*(A+B)**2<=1.2*
SQRT(A—B)/0.2+14,

если A=6, B=2. Для большей наглядности промежуточные результаты в порядке их получения будем обозначать P1, P2, P3 ... и т. д.

Тогда P1=A+B=8; P2=A—B=4;
P3=$\sqrt{P2}$=2;
P4=P1**2=64; P5=—A=—6;
P6=P5/B=—3; P7=0.25*P4=16;
P8=1.2*P3=2.4; P9=P8/0.2=12;
P10=P6+P7=13; P11=P9+14=26;
P12=(P10<=P11)='1'.

Таким образом, при A=6, B=2 рассматриваемое выражение имеет значение '1' (истина).

Структура ПВ-программы

Программа на языке ПВ включает в себя заголовок, перечень аргументов и результатов, тело программы и входной поток данных. При этом перечень аргументов и результатов и входной поток данных не являются обязательными и могут отсутствовать.

Заголовок является обязательной частью ПВ-программы и имеет структуру

.АЛГ. <имя>;

или

.АЛГ. <имя>(<список аргументов и результатов с указанием типов>).

В угловых скобках будем записывать изменяющиеся конструкции, которые в каждом конкретном случае принимают некоторые конкретные значения.

⟨имя⟩ — идентификатор, содержащий не более 7 символов и являющийся именем (обозначением) ПВ-программы;

⟨список аргументов и результатов с указанием типов⟩ — перечень описаний типов аргументов и результатов, в общем случае может быть представлен в следующем виде:

⟨описание переменных одного типа⟩ или

⟨описание переменных одного типа⟩, ⟨список аргументов и результатов с указанием типов⟩;

⟨описание переменных одного типа⟩ есть ⟨указатель типа⟩, ⟨список переменных⟩;

⟨список переменных⟩ есть ⟨идентификатор переменной⟩, или ⟨идентификатор переменной⟩, ⟨список переменных⟩.

Указателем типа может быть одно из следующих ключевых слов: .БУЛ. или .ЛОГ.— булевский (логический); .БУК. (⟨целое число⟩) или .ЛИТ. (⟨целое число⟩) — символьный с длиной ⟨целое число⟩ символов; .ВЕЩ. или .ДЕЙ.— вещественный (действительный) с точностью представления 6 значащих цифр; .КОМ.— комплексный; .МЕТ.— меточный; .НАТ.— натуральный с представлением чисел до $2^{15}-1=32767$ включительно; .УПР.— управляющий; .ЦЕЛ.— целый с представлением чисел от —99 999 до +99 999.

При необходимости количество разрядов, которое должно быть обеспечено представлением чисел в памяти ЭВМ для переменных вещественного (действительного), натурального или целого типов, может быть указано явно в виде целого числа, заключенного в скобки, например: .ВЕЩ. (8) — вещественный тип с точностью представления 8 значащих цифр, .ЦЕЛ. (7) — целый тип с 7 значащими цифрами, т. е. от —9 999 999 до +9 999 999, .НАТ. (20) — с представлением чисел до $2^{20}-1=1\,048\,575$ включительно. Максимальное число разрядов для вещественных чисел равно 16, для целых — 15, для натуральных — 31 (двоичный) разряд.

При описании таблиц (массивов) после указателя типа записывается ключевое слово .ТАБ. (таблица), а в списке переменных после идентификатора соответствующей переменной в квадратных скобках записывается наименьшее и наибольшее значения (разделяемые двоеточием) по каждой размерности таблицы (по каждому индексу массива). Так, например, для задания линейной таблицы (одномерного массива) целых чисел A с изменением индекса от 0 до 99 и прямоугольной таблицы (двумерного массива) действительных чисел B, состоящей из 10 строк и 5 столбцов, пронумерованных по порядку, следует записать

.ЦЕЛ..ТАБ. A[0:99], .ВЕЩ..ТАБ. B[1:10, 1:5]

В заключение приведем пример полного заголовка ПВ-программы, предназначенной для вычисления значений прямоугольной таблицы T по формуле

$$T_{i,k}=X_k+M\cdot(i-k),$$

где X=18,34; 19,56; 7,04; 83,17; 0,74;
M=12358,647; i, k=1, 2 3, 4, 5.

Если дать программе имя A21, то заголовок может быть записан в виде

.АЛГ. A21(.ВЕЩ..ТАБ. X[1:5], .ВЕЩ.(8)M, .ВЕЩ.(8) .ТАБ. T[1:5, 1:5], .ЦЕЛ. I ,K);

Перечень аргументов и результатов является необязательной частью ПВ-программы и служит для увеличения ее наглядности.

Список идентификаторов аргументов записывается после ключевого слова .АРГ. (аргументы), а список идентификаторов результатов работы алгоритма располагается после ключевого слова .РЕЗ. (результаты). Оба они должны завершаться символом ;. Так, для программы A21 перечень аргументов и результатов выглядит так:

.АРГ. X, M, I, K; .РЕЗ. T;

Тело ПВ-программы начинается с ключевого слова .НАЧ. (начало) и завершается ключевым словом .КОН. (конец). После .НАЧ. при необходимости записываются описания промежуточных (ра-

бочих) переменных. Таких описаний может быть несколько, одно или ни одного. Тело программы должно содержать хотя бы один оператор.

Описания переменных имеют вид

⟨указатель типа⟩⟨список переменных⟩;

где ⟨указатель типа⟩ и ⟨список переменных⟩ в точности совпадают с аналогичными конструкциями, рассмотренными выше.

Операторы являются основным содержанием ПВ-программы, они определяют операции и порядок их выполнения. Операторы записываются последовательно и разделяются символом ;. Последовательность операторов, выполняющихся в процессе исполнения ПВ-программы один за другим, называется серией. Для общности положим, что серия может включать в себя произвольное количество операторов: ни одного, один, два и т. д.

ПВ-программа записывается в позициях 2—72 стандартного бланка для записи программ. При отсутствии бланков ПВ-программа может быть записана на любом листе бумаги, но при этом нельзя опускать символы пробелов, которые должны обозначаться ␣; обязательно ставить пробел в начале каждой строки. Кроме того, необходимо тщательно следить, чтобы общее количество символов в каждой строке не превосходило 72.

Входной поток. Если в ПВ-программе имеется хотя бы один оператор ввода (см. ниже), то после тела программы, т. е. после ключевого слова .КОН., записывается входной поток.

Он начинается ключевым словом .ИСХ. (исходные данные). После него в той же или в следующей строке записываются вводимые данные (значения величин в соответствии с порядком их следования в операторах ввода). Данные разделяются пробелами или запятыми, окруженными любым количеством пробелов. Данные символьного и логического типов открываются и закрываются апострофами. В конце входного потока никакие знаки препинания не ставятся, он завершается ключевым словом .КОН..

Исходные данные, относящиеся к одному оператору ввода, могут располагаться в одной или нескольких строках, в одной строке могут располагаться данные, относящиеся к нескольким операторам ввода.

Пусть, например, A — линейная таблица, описанная как .ВЕЩ. A[1:5], B и C — переменные, описанные как .ЦЕЛ.B, .ВЕЩ.C; они должны получить следующие значения:

A=—0,54; 0,003; 1,356; 2,074; 7,385;
B=1024; C=—2,374 · 10^{-6}.

В этом случае оператор .ВВОД.(A, B, C) обусловит такое описание входного потока:

.ИСХ.
—0,54␣,␣0.003␣␣1,356␣␣2.074␣
7.385␣␣1024␣␣—2.374E—6␣␣
.КОН.

Операторы языка ПВ подразделяются на простые и составные. Простые операторы непосредственно осуществляют переработку информации, составные включают в себя те или иные простые операторы и лишь управляют последовательностью их выполнения.

К простым операторам в языке ПВ относятся операторы присваивания, ввода и вывода.

Оператор присваивания служит для вычисления значений выражений, преобразования этих значений к требуемой форме и присваивания результатов переменным и элементам таблиц.

В общем случае оператор присваивания имеет вид

X:=B;

где X — переменная, элемент таблицы или имя таблицы, B — выражение соответствующего типа. В операторах присваивания допускается использование идентификаторов таблиц, но при этом все такие таблицы должны иметь одинаковые описания размерности.

Если выражение не помещается в одной строке, его можно разорвать в любом месте и продолжить запись, начиная с любого места следующей строки. Если разрывается некоторая текстовая константа, имеющая вид 'A...BC...X', после символа B, то в первой строке записывается (с закрывающим апострофом) первая часть текстовой константы 'A...B', а вторая строка начинается с операции соединения строк,

после которой записывается (с открывающим апострофом) оставшаяся часть текстовой константы, т. е. начало следующей строки имеет вид !!'C...X'. Разрывать ключевые слова при переходе со строки на строку запрещается.

Оператор ввода предназначен для считывания информации из входного потока, преобразования ее в требуемую форму и присваивания полученных результатов переменным, элементам таблиц и таблицам, указанным в операторе.

В общем виде оператор ввода имеет структуру

.ВВОД. (⟨список ввода⟩);

В списке ввода могут стоять идентификаторы переменных, таблиц и отдельные элементы таблиц, разделенные запятой. Указание имени таблицы в списке ввода эквивалентно перечислению всех элементов таблицы в порядке их следования (для прямоугольных таблиц — в порядке их следования по строкам). При выполнении оператора ввода данные из входного потока в порядке их следования в потоке присваиваются элементами списка ввода в порядке их перечисления в списке. Оператор ввода завершает свою работу после окончания обслуживания последнего элемента списка ввода. При следующем выполнении того же самого или другого оператора ввода входной поток продолжает рассматриваться с того места, на котором закончилось его рассмотрение раньше. Если входной поток заканчивается раньше, чем список ввода, то происходит аварийный останов (прерывание выполнения программы).

Оператор ввода предназначен для преобразования информации к виду, удобному для восприятия человеком, и вывода ее на алфавитно-цифровое печатающее устройство (АЦПУ) ЭВМ.

Оператор вывода имеет структуру

.ВЫВ. (⟨список вывода⟩);

В списке могут находиться имена переменных, элементы таблиц, имена таблиц, различные выражения, в том числе константы (включая текстовые константы, т. е. последовательности символов, заключенные в апострофы). Отдельные элементы списка разделяются запятыми.

При выполнении оператора последовательно печатаются значения элементов списка вывода. Если эти значения занимают более одной строки АЦПУ (более 120 символов), то автоматически начинается печать следующей строки. Выводимые значения не разделяются пробелами, за исключением целых или натуральных чисел; перед первыми добавляется два пробела, перед вторыми — три. При выводе арифметических данных одна позиция резервируется для знака; в ней для отрицательных чисел ставится знак «минус», для положительных — пробел.

Для организации печати с новой строки ставится оператор перехода к новой строке — ключевое слово .СТР. (строка). Допускается объединение операторов вывода и перехода к новой строке:

.ВЫВ..СТР. (⟨список вывода⟩);

или

.СТР..ВЫВ. (⟨список вывода⟩);

Для сокращения записи списка вывода можно перед текстовыми константами записывать коэффициент повторения (целое число, заключенное в скобки). Так, вместо — — — — — — — — — — можно употребить запись (10)—. Например, операторы вывода

.СТР..ВЫВ.((19)'␣','ЭЛЕКТРОДВИГАТЕЛЬ ЭК283');

.СТР..СТР..ВЫВ.((22)'␣', 'РАСЧЕТ МОЩНОСТИ');

.ВЫВ..СТР.((22)'␣',(15)'=');

обеспечивают вывод на печать с новой строки 19 пробелов, затем текста ЭЛЕКТРОДВИГАТЕЛЬ ЭК283, пропуск одной строки и печать после 22 пробелов текста РАСЧЕТ МОЩНОСТИ, который подчеркивается (в следующей строке) линией, состоящей из 15 символов = .

Окончание следует.

Е. ГЕЛЬТИЩЕВА, Г. СЕЛИХОВА

Режим работы за дисплеем

Сейчас компьютеры широко внедряются в учебный процесс. В школах оборудуются дисплейные классы (ДК) и кабинеты вычислительной техники (КВТ), оснащенные мини-ЭВМ и микро-ЭВМ.

Задача исследования — изучить влияние различных режимов работы учащихся на ЭВМ в общеобразовательной школе. Цель — обосновать оптимальное время работы школьников на дисплеях.

На базовом объекте — общеобразовательной школе № 542 под наблюдением находились учащиеся IX—X классов, выполнявшие работы на ЭВМ по программированию.

Дисплейный класс был организован в помещении столярной мастерской и оборудован 16 дисплеями «Электроника 15 ИЭ-00-13» для каждого учащегося.

Режим учебных занятий был построен так: начало учебных занятий в школе в 9 ч 20 мин, учащиеся девятых классов после трех уроков, а учащиеся десятых классов после четырех уроков имели одночасовой перерыв на обед.

Занятия по основам информатики и вычислительной техники строились следующим образом: в начале урока педагог проводил объяснение учебного материала, а учащиеся сидели за столами, после этого им разрешалось работать на дисплеях и они пересаживались на рабочие места, оборудованные ЭВМ.

Физиологические и психологические исследования осуществлялись до и после учебных занятий. Время работы учащихся на ЭВМ колебалось от 5 до 40 мин. Под наблюдением находилось 50 здоровых учащихся (I и II группа здоровья), имевшие нормальный слух и зрение. Один школьник проходил от 1 до 3 проверок. Всего было выполнено более 2000 исследований.

Как показали наблюдения, работа старшеклассников за дисплеем характеризуется большим нервно-эмоциональным и зрительным напряжением. Плотность работы за дисплеем высокая. При выделении времени для работы на ЭВМ учащиеся работают не отвлекаясь. Работа сопряжена и с положительными, и с отрицательными эмоциями. Для оценки функционального состояния были выбраны наиболее информативные методики, позволяющие оценить состояние центральной нервной системы и зрительного анализатора: хронорефлексометрия на световой и звуковой раздражители; определение быстроты восприятия и переработки информации, концентрации внимания по модифицированным тестам Крепелина; критическая частота слияния световых мельканий (КЧССМ); определение ближайшей и дальней точек видения по методике С. Л. Шаповалова с последующим вычислением коэффициента утомляемости зрительного анализатора. Последовательность проведения физиологических исследований была одинаковой как до занятий, так и после них.

При проведении физиологических и психологических исследований учитывалось самочувствие учащихся.

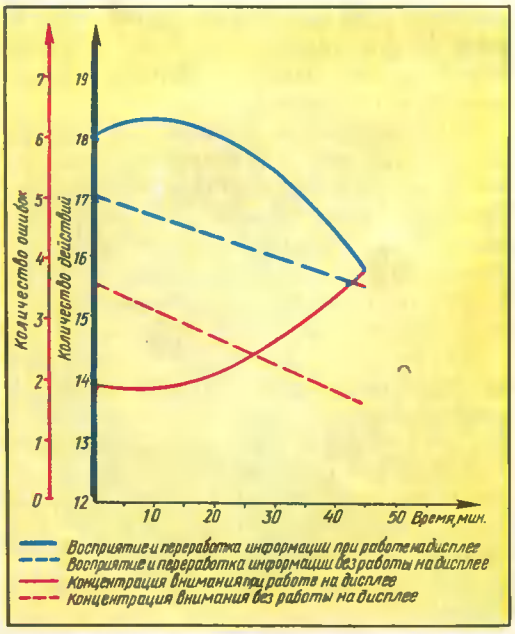
Те же учащиеся в том же дисплейном классе проходили контрольные исследования в те моменты, когда они не работали за дисплеями.

Обработка результатов осуществлялась методом корреляционно-регрессионного анализа. Получены математические модели в виде уравнений регрессии второго порядка, характеризующие изменение функционального состояния учащихся в зависимости от времени работы за дисплеем.

Проверка значимости коэффициентов регрессии осуществлялась по t-критерию. Проверка согласия модели экспериментальным данным по F-критерию.

На рисунках представлены графические зависимости влияния временного фактора на отдельные показатели функционального состояния учащихся, построенные по полученным моделям.

Анализ зависимости функционального состояния по изучаемым показателям при работе школьников на дисплеях и без работы на них (контроль) показывает ухудшение функционального состояния с увеличением времени работы на дисплеях.



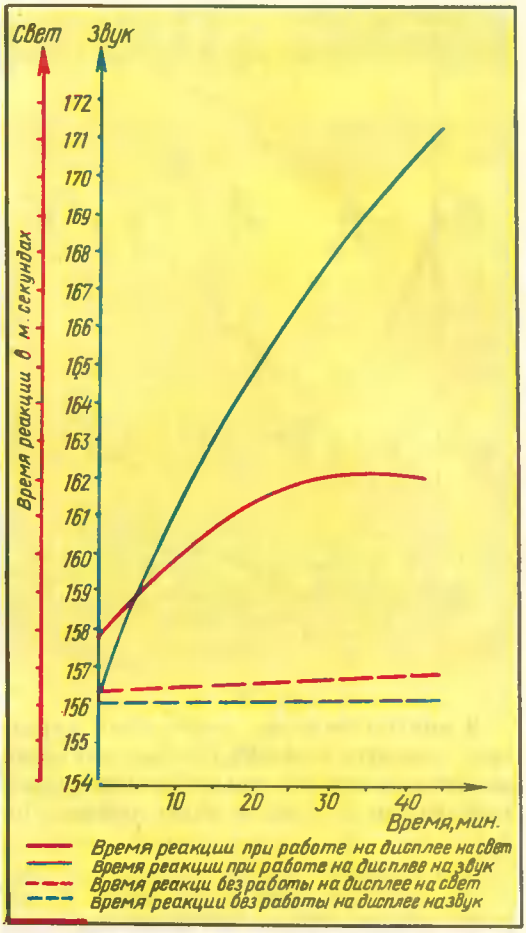
1

На рис. 1 видно, что, начиная с 20 мин работы на дисплеях, ухудшается функциональное состояние высшей нервной деятельности, выражающееся в снижении восприятия и переработки информации, концентрации внимания. При этом ее изменения выражены менее значительно по сравнению с восприятием и переработкой информации.

В контрольных исследованиях функция восприятия и переработки информации снижалась, но была менее выражена по сравнению с работой учащихся за дисплеями, в то время как функция концентрации внимания улучшалась, т. е. имела противоположную направленность.

Скрытое время рефлекторных двигательных реакций на световой и звуковой раздражители (рис. 2) свидетельствует о нарушении взаимодействия между зрительными и слуховыми анализаторами. Несущественные проявления нарушений взаимодействия отмечаются уже при работе на дисплеях в течение 10 мин, допустимые — при работе 20—25 мин и выраженные при работе на дисплеях в течение 30 мин и более.

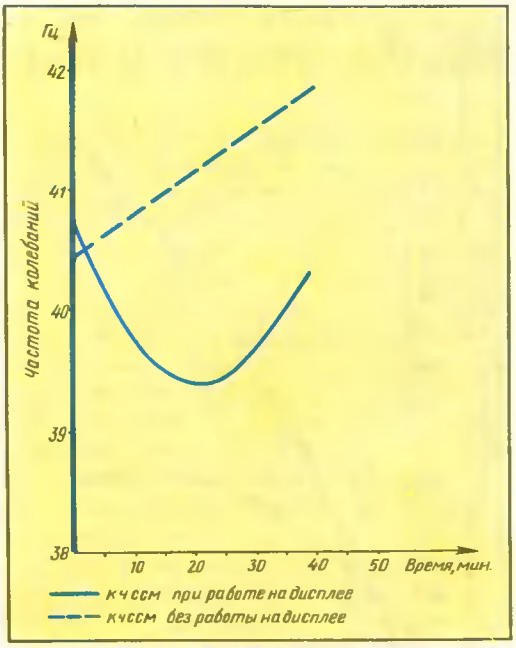
В контрольных исследованиях скрытое время рефлекторных реакций на свет



2

товой и звуковой раздражители почти не изменялось.

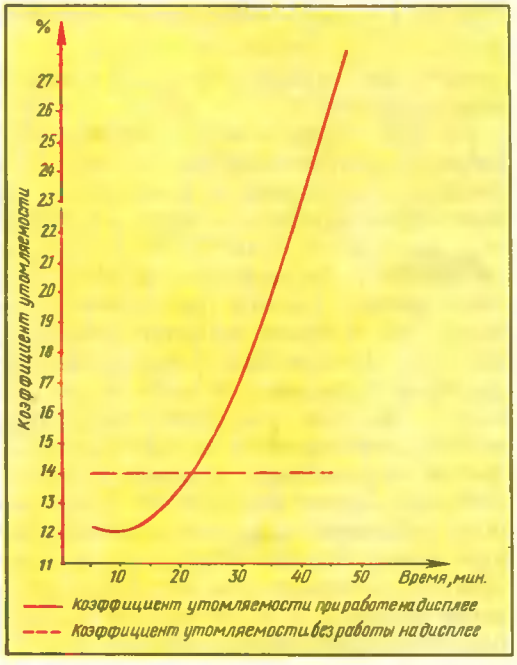
На рис. 3 представлена динамика изменения критической частоты слияния световых мельканий в зависимости от длительности работы на дисплеях. В первые минуты (до 20—25 мин) работы на дисплеях уменьшается КЧССМ, а в последующем разница между показателями до и после занятий сглаживается, что может быть обусловлено характером выполняемой работы на дисплеях. Вначале учащиеся интенсивно включаются в работу и набирают данные на клавиатуре, контролируя результаты на экране дисплея. В последующем, особенно при неправильном получении ответа, вносят коррективы в саму программу, анализируя, где были допущены ошибки, и темп работы на дисплее несколько снижается.



3

В контрольных исследованиях характер изменения КЧССМ был противоположным тем, которые были получены при работе учащихся на дисплеях. По-

4



казатели КЧССМ к концу урока улучшались. Но такая направленность изменений может означать первую стадию развития утомления.

Более выраженные сдвиги со стороны функционального состояния зрительного анализатора были получены при определении ближайшей и дальней точек измерения видимости и вычислении коэффициента утомляемости, прямо указывающего на развитие утомления зрительного анализатора.

На рис. 4 представлено изменение коэффициента утомляемости в зависимости от времени работы на дисплее и без нее. Как видно из рисунка, первые сдвиги в сторону ухудшения функционального состояния зрительного анализатора проявляются уже через 15 мин работы, допустимые — через 20—25 мин и существенные — через 30 мин работы и более.

Физиологические и психологические исследования подтверждают анализ субъективных данных о самочувствии учащихся. Так, уже через 20—25 мин работы на дисплеях некоторые учащиеся жаловались на общую усталость. Начиная с 30 минуты работы на дисплеях количество таких учащихся увеличилось в 3 раза, а при работе в течение 35—40 мин — в 9 раз. Жалобы на усталость глаз отмечались учащимися в основном при работе на дисплеях в течение 30 мин и более.

В итоге при организации учебных занятий по основам информатики и вычислительной техники и работе учащихся на дисплеях следует рекомендовать:

1. Работу учащихся на дисплеях проводить во второй половине урока, после объяснения учителем теоретического материала.

2. Длительность работы на дисплеях должна быть ограничена 20—25 мин во время урока.

3. Положительно следует оценить организацию одночасового перерыва для обеда и отдыха в середине учебного дня.

Такой режим можно рекомендовать не только в общеобразовательных школах, но также в межшкольных учебно-производственных комбинатах.

А. ШМЕЛЕВ
Канд. психол. наук

ДЕТСКАЯ БОЛЕЗНЬ КОМПЬЮТЕРНОГО ВСЕОБУЧА

С появлением персональных учебных микро-ЭВМ в наших школах и учебных производственных центрах появились и компьютерные игры. При недостатке русскоязычных обучающих программ (по обычным предметам да и по информатике тоже) этот дефицит программного обеспечения восполняется компьютерными видеоиграми — в подавляющем большинстве случаев зарубежного производства — с англоязычными инструкциями, с характерным для западного общества подбором сюжетов. Предназначенные прежде всего для рекламы компьютерной техники, эти игры не ставят перед пользователем интеллектуальных задач, рассчитаны на броский внешний эффект: они призваны создать у начинающего пользователя убежденность в простоте обращения с компьютером, поразить его воображение богатством красок, графических форм, звуковых эффектов. Для рекламной игры, конечно, необходимыми атрибутами является предельная простота, очевидность игровых условий. Цели этих развлекательных игр даны в непосредственной, чувственно-наглядной форме — надо догонять, убегать, уклоняться, прыгать, ловить, стрелять, управляя поведением экранного автогероя с помощью небольшого числа особых клавиш, обычно отвечающих в диалоговых программах за перемещения курсора. В ходе игры пользователь получает непрерывную информацию о своих результатах (текущая сумма очков), ресурсах (запас горючего в баке, патронов, «жизней»), о кульминационных успехах и неудачах (и то и другое сопровождается впечатляющими зрительными и шумовыми эффектами). По мере увлечения игрой очень быстро возникает своеобразная иллюзия сверхвключенности, своего рода квазисуществования в особом мире, в силу этого притягательная сила компьютерных игр во много раз превосходит и без того давно тревожащую педагогов привлекательность обычного кинофильма или телепередачи. Возникает опасность наркотизирующего эффекта, от которой прежде всего не защищена неустойчивая психика ребенка.

В этой статье мы хотели бы поставить вопрос об ответственности специалистов — педагогов и психологов — за распространение и применение компь-

ютерных игр в школах. А также, по-видимому, и за их применение в сети так называемых досуговых центров, создание которых в массовых масштабах уже запланировано Министерством культуры СССР на ближайшую пятилетку.

Сотни тысяч (а может быть, сейчас уже и миллионы) юных операторов и программистов захвачены настоящей эпидемией компьютерных игр, а ученые-педагоги и психологи все еще оперируют понятиями времен программированных учебников.

Сказывается и наличие пресловутого психологического барьера перед ЭВМ у специалистов-гуманитариев. О легкости снятия этого барьера и говорят прежде всего сторонники применения компьютерных игр на первых этапах обучения компьютерной грамотности. Но в наших условиях уже сегодня становится очевидным своеобразный парадокс*: у детей нет вообще какого-то особого барьера, который надо было бы снимать сверхувлекательными играми (как и нет в большинстве случаев недостатка мотивации на занятия с компьютером), а у «озабоченных» взрослых игры вызывают предубежденное отношение как к «пустой трате времени» — на сегодня почти не существует игр, насыщенных обучающим материалом в такой концентрации, которая необходима для взрослых. Реальной становится опасность, что появится пропасть между «серьезными» взрослыми, не играющими в игры, и «играющими» детьми — двумя социальными группами, не способными понять внутренний мир друг друга. Для преодоления этого разрыва нужны реальные исследования компьютерных игр, в том числе из позиции включенного наблюдателя, способного испытать на себе эффект воздействия различных этапов освоения существующих компьютерных игр.

Безусловно, такое включенное наблюдение должно быть дополнено и анализом литературы, и специальными объективными экспериментами.

Опираясь на зарубежную и небогатую отечественную литературу по компьютерным играм, а также на результаты самостоятельного анализа проблемы, мы считаем возможным предложить следующую первичную классификацию функций компьютерной игры и взаимосвязанных с этими функциями проблем.

1. Социально-имитационная функция. Когда компьютеризация идет сверху под определенным административным нажимом, то у некомпетентного исполнителя возникает искушение создать видимость массовых успехов компьютерного всеобуча — масса начинающих игроманов, рвущихся к дисплеям, выдается (и по невежеству воспринимается) за серьезно интересующихся компьютерами учеников. Складываются отношения своего рода двойного паразитирования: невежественная администрация паразитирует на потребительских настроениях несознательных подростков и юношей, не желающих учиться, но жаждущих развлекаться у компьютера. Понятна мнимость успехов от подобного бесконтрольного применения компьютерных игр развлекательного характера: у ребят со слабой мотивацией учения последняя не укрепляется, а заглушается нарастающей азартной страстью к игре.

2. Рекреативная функция. Умеренное

* Автор статьи опирается в констатации этого парадокса не только на литературу, но и на собственные наблюдения над первыми шагами приобщения к ЭВМ школьников (в кружках), студентов (в компьютерном практикуме), зрелых преподавателей (занятия ФПК МГУ).

использование развлекательных видеоигр само по себе может быть безвредным и даже полезным в особом смысле — в смысле того разгружающего, восстанавливающего (рекреативного) эффекта, который оказывает игра на человека, чрезмерно озабоченного, напряженного, тревожного. Оправдано применение компьютерных игр в кабинетах психологической разгрузки на промышленных предприятиях и в учреждениях. Но существенная для взрослых, эта функция не столь важна для детей: взрослые хотят в развлечениях снять напряжение, переключить внимание с более тревожных обстоятельств на безобидную игру, для детей же игры выполняют скорее противоположную функцию — они ищут в них источник повышенного напряжения, риска, ситуаций испытания, преодоления.

3. Функция самоиспытания. Человек может удовлетворять в игре потребность самопознания — получить информацию о себе, о границах и не использованных им пока резервах в развитии своих способностей. Эта мотивация самодиагностики в случае ее осознания входит в психологическую структуру игры как особого психотехнического действия по осознанному упражнению собственных психических функций. Некоторые игры не имеют четкого предметного содержания и непонятны специалистам-предметникам — представителям наук о внешнем окружающем человека мире. Но они должны быть понятны педагогам и психологам: в ходе такой игры человек не получает новой информации о внешнем мире, но получает необходимую ему информацию о себе самом — о том, как лично он может взаимодействовать с определенной моделью уже известного ему (но из позиции пассивного внешнего наблюдателя) локуса внешнего мира. Например, каждый школьник хотя бы теоретически знает, что для объезда препятствия справа руль автомобиля надо поворачивать вправо, но далеко не каждый знает, может ли он осуществить это известное ему требуемое действие в ограниченный отрезок времени. Такую возможность дают так называемые локомоционные компьютерные игры («Ралли», «Управление самолетом», «Слалом» и т. п.).

Компьютер дает для самодиагностики важные преимущества:

конфиденциальность, — пользователь может получить информацию о себе, не раскрывая ее другим людям;

возможность широкого социального сравнения — если в памяти ЭВМ накоплена «таблица рекордов» по представительной выборке игроков, то пользователь получает возможность сравнить себя с большим числом людей, выполняющих аналогичную деятельность;

имитация: видеоигра имитирует реальную деятельность, но себестоимость ошибок обходится игроку гораздо «дешевле»;

игровой характер самопознания — самопознание соединяется с развлечением;

оперативная компенсация — при обнаружении слабых результатов игрок может тут же приступить к тренировке соответствующего качества, т. е. самопознание соединяется с тренингом.

Но одновременно с преимуществами указанная слитность в компьютерной игре элементов самопознания с элементами развлечения и тренинга создает специальную проблему — трудность осмысления игроком диагностической сути игры. Программа типичной игры имитирует, как правило, синтетические ситуации и действия и редко требует активизации только одной психологической функции. Не осознавая, что именно тестируется, игрок не может построить психотехнического действия.

В настоящий момент стоит проблема создания «диагностически чистых» компьютерных игр, или, лучше сказать, компьютерных диагностических методик, реализованных в форме игры.

4. Функция психотренинга. В отличие от учебных тренажеров с игровыми элементами, прямо направленных на закрепление конкретных знаний и навыков, компьютерные игры часто апеллируют к общим психическим свойствам — настолько общим, что нам представляется удобным описывать эти свойства на языке черт индивидуальности. Когда игрок упражняется в принятии мгновен-

ных решений в скоростной игре на компьютере, он при этом не учится принятию решений в каких-то конкретных ситуациях, ибо реальная ситуация может значительно отличаться от условной игровой ситуации. Но тем не менее при этом тренируются такие универсальные качества, как решительность (скорость преодоления сомнений и колебаний), настойчивость (готовность продолжать попытки, несмотря на неудачи), ответственность и т. п. В каком смысле ответственность? В психологической науке это понятие связано с понятием локуса контроля, которое исследовалось как теоретически, так и экспериментально. Показано, что разные ситуации деятельности объективно различаются по соотношению внутреннего и внешнего локуса контроля: если результат зависит от внешних и случайных обстоятельств, говорят о внешнем контроле, если — от самого человека, говорят о внутреннем. С этой точки зрения, безответственность рассматривается как привычка к внешнему контролю: даже тогда, когда человек может и должен самостоятельно контролировать ситуацию, он снимает с себя ответственность — перекладывает ее на других людей или на безличные обстоятельства. И в результате не мобилизуется тогда, когда сам может и должен быть мобилизован. Компьютерные игры, предполагающие потенциальное наличие у игрока какой-то единственной выигрышной стратегии, мобилизуют его на то, чтобы самостоятельно отыскивать эту стратегию, т. е. приучают к деятельности с внутренним контролем, воспитывают в какой-то мере такое качество, как ответственность за свои поступки.

Но к сожалению, указанный самовоспитательный потенциал большинство компьютерных игр обнаруживают далеко не всегда. И это зависит в решающей степени не от самой игры, а от человека! Это зависит от того, какой мотив толкает человека к игре. Педагогам необходимо научиться различать два мотива, о которых мы уже писали в литературе: это мотив тренинга и мотив псевдокомпенсации. Если первый — основа, неосознаваемая предпосылка будущего сознательного психотехнического тренинга (аутопсихотренинга), то второй мотив выражает стремление человека к поиску таких искусственных условий, в которых его неадекватные, но привычные и желательные способы поведения смогут быть осуществлены. Во втором случае игра способствует не столько подготовке к столкновению с реальностью, сколько потворствует бегству от реальности. Она дает человеку возможность квазисуществования в псевдомире, в котором царит логика поправимых ошибок: можно многократно разбиваться в гоночном автомобиле, в самолете, быть проглоченным людоедом и т. п., но каждый раз с перезапуском программы вновь оказываться возрожденным и невредимым.

Произвольность в выборе любимой игры создает перед человеком с мотивом псевдокомпенсации искушение однобокого тренинга: он совершенствуется в чем-то одном, что у него получается лучше других, и избегает тех задач, где он менее успевает.

Игровой психотренинг на компьютере — дело тонкое и небезопасное, оно может привести к отрицательным результатам. Для того чтобы этого избежать, психологи должны еще основательно потрудиться в своих лабораториях над разработкой типовых программ компьютерного психотренинга — с учетом индивидуальных особенностей определенных категорий людей. Это относится и к детям, и ко взрослым. Но в отношении детей задача, несомненно, является особенно актуальной, ответственной и сложной. Тем более что, если резко ограничить доступ детей к играм, положение заведомо ухудшится (по известному механизму запретного плода).

5. Функция соревнования. Мы уже говорили о том, что компьютерные игры предоставляют исключительно широкие возможности для социального сравнения (себя с другими). Они могут стать средством удовлетворения потребности в самоутверждении, доказательстве своего превосходства, очень часто они вызывают соревновательную мотивацию у людей, казалось бы далеко не спортивных. В этом смысле компьютерные игры представляют собой особый вид

спорта, и не исключено, что в самое ближайшее время мы станем свидетелями стихийного появления международных чемпионатов (например, по игре «Приключения в пещере») и международных федераций. В этом смысле компьютерные игры несут с собой опасность узкой специализации и дисгармоничного развития,— но в данном случае не в области физического, а в области психологического развития человека, что, несомненно, гораздо опаснее. Если этим тенденциям суждено будет осуществиться, то дело педагогов и психологов — как можно быстрее учредить своеобразное компьютерное психотехническое многоборье. Чтобы у игрока, развившего поразительную ловкость пальцев и преуспевшего в психомоторике, не возникало неадекватного самомнения, если при этом он не может обыграть элементарную шахматную программу или решить какую-либо комбинаторную головоломку.

Известно, какую важную роль сыграли и продолжают играть комплексы ГТО в области физического воспитания подрастающего поколения в нашей стране. Сейчас на повестке дня вопрос о расширении этого комплекса и о включении в его состав специально отобранных компьютерных игр-тестов. Научное планирование, прогнозирование и контроль должны опережать в этой области стремительно развивающуюся реальность. В этом состоит ответственность ученых.

6. Функция образования (просвещения). Хотя большинство игр не несут особой информации, не дают человеку новых знаний, а опираются на уже известные ему сюжеты и закономерности, тем не менее нецелесообразно было бы полностью игнорировать эту скорее потенциальную, чем осуществившуюся функцию компьютерных игр. В этой области ученым еще предстоит работа — над созданием принципиальных моделей игр, в которых достижение каких-то чисто игровых целей (суммы очков) опосредовано усвоением некоторой предметной информации. Если в сегодняшних психомоторных играх ребенок помогает персонажу пальцами (указывает, куда бежать, когда стрелять и т. п.),

то в будущем надо подумать о конструировании игр, в которых ребенок сможет оказывать автогерою помощь своими знаниями — объясняя ему определенные закономерности и т. п. Возможно конструирование и особых гибридов из учебных и кооперативных игр: один ученик непосредственно идентифицируется с персонажем (управляет им пальцами), а другой просвещает этого персонажа (фактически своего партнера); затем, по отношению к новому разделу учебной программы, роли меняются.

7. Мотивационное управление учебной деятельностью. Эта особая функция в своем примитивном исполнении выражается формулой: «Ну, дети, если выполните учебное задание, получите возможность поиграть с машиной». Как показали специальные весьма убедительные исследования психологов, назначение внешней награды за интеллектуальный труд слишком часто приводит к утрате внутреннего интереса к этому труду. В таком случае мы навязываем невольно ребенку такую логику: «Мы, конечно, понимаем, деточка, что учеба — скучное и подневольное занятие, но зато потом ты получишь удовольствие, заработав игру». Чем меньше игра связана по содержанию с изучаемым предметом, тем сильнее эта опасность.

Противоположный вариант — игра впереди учения. В этом случае она призвана заинтересовать, увлечь ученика. Но здесь она обязательно должна

нести в себе какие-то существенные признаки изучаемого предмета. С таким применением компьютерных игр связаны надежды их сторонников при обучении информатике и программированию: по замыслу вначале ребенок должен заинтересоваться игрой, а затем вносить в нее модификации, т. е. заинтересоваться тем, как игра сделана, и тем самым фактически приступить к конструированию программ. Но такой сдвиг мотивации происходит далеко не автоматически и, к сожалению, далеко не всегда. Он самопроизвольно наступает у детей с уже развитой конструкторской «жилкой». С меньшей гарантией — у детей с незаглушенной еще природной любознательностью. А вот у детей с невротическими симптомами, эмоционально дезадаптированных (с трудностями общения в коллективе, с отставанием в учебе, с остро переживаемыми конфликтами в семье) такой переход может и не произойти вовсе: этап игры только укрепит мотив псевдокомпенсации — ребенок с аутическим радикалом (замкнутый на свои фантазии) еще сильнее аутизируется, слабые симптомы враждебной агрессивности по отношению к окружающим могут перерасти в настоящие садистские наклонности и т. п. То есть мы видим, что подобный прием следует применять строго избирательно, тщательно продумывая набор игр. Иначе вместо личностного и познавательного развития детская психика регрессирует. Не следует перебарщивать, стараться поразить воображение ребенка. Вместо того чтобы обрушивать на ребенка игры под девизом «Убей их всех!» («Космическая война», «Супер-кобра», «Хариер», «Королевская пещера», «Али-баба» и др.), достаточно просто предоставить возможность ребенку для графического интерактивного диалога. Ведь даже в старшем школьном возрасте подросткам бывает просто интересно сначала только нажимать на клавиши и управлять геометрическими узорами на экране. На этом основан психолого-педагогический потенциал известного учебного языка-интерпретатора Лого.

По другим предметам (кроме информатики) игры также могут оказаться полезными на первом этапе — для экспозиции проблемы. Игра позволяет быстрее погрузить ребенка в ту проблемную среду, которая обеспечивает необходимую чувствительность учащегося (восприимчивость) к учебной информации.

8. Функция объекта конструктивной деятельности.

Сама по себе программа несложной компьютерной игры может быть преподнесена в качестве объекта для манипуляций — как некий информационный конструктор. Большинство педагогов-практиков считают такой вариант использования игр чуть ли не безупречным (имеется в виду, конечно, прежде всего обучение программированию). Но и здесь недооцениваются скрытые опасности. Назовем прежде всего ставший уже широко известным феномен одержимого программиста — хакера. Оказывается, и увлечение конструированием, будучи превращенное в самоценное занятие, приводит к дисгармоничному развитию человека. Первый признак хакерства — безразличие ко всему, что не имеет непосредственного отношения к сидению за дисплеем. Хакеры могут быть отличными работниками, но их глубокая ограниченность, невосприимчивость к гуманитарным ценностям неизбежно накладывают отпечаток «технократического «кретинизма» и на их продукцию. В хакерстве часто проявляется та же самая симптоматика бегства от реальности, что и у игроков-потребителей. Разница лишь в том, что в отличие от потребителей, одержимые конструкторы компьютерных игр создают сами для себя (и для окружающих) искусственные миры, по отношению к которым они переживают своего рода упоение властью, чувствуя себя творцами некоего псевдобытия.

По нашему глубокому убеждению, конструирование игр должно быть с самого начала подчинено некой социально осмысленной цели, должно носить прикладной характер. Один из вариантов — конструирование игр для психодиагностики и психологического тренинга — апробируется сейчас автором статьи в кружковой работе со школьниками седьмых классов.

9. Имитационный тренинг. Иногда

довольно сложно провести четкую грань между психологическим и имитационным тренингом — между программами-играми и компьютерными тренажерами. Где-то на границе между двумя этими широкими классами прикладных программ находятся имитационные игры. По своим способам исполнения имитационные игры бывают практически тождественными психотренинговым играм. Разница — в степени приближения к некоторым реальным ситуациям деятельности. В той степени, в какой игрок приучается определять на слух по гулу «мотора», на каких оборотах работает двигатель автомобиля (и моменты, когда нужно переключать передачу), в той степени такие игры как «Супер-ралли» на ПК «Ямаха» можно считать имитационными. Такие игры позволяют готовиться к конкретному виду деятельности.

Их можно рекомендовать на первых этапах игрового тренинга, ибо по отношению к этим играм проще актуализировать и укрепить именно тренинговую мотивацию (по сравнению с конкурирующей мотивацией псевдокомпенсации). На имитационных играх проще отладить переход от компьютера — к реальности: натренировав на компьютере какой-то умственный или моторный навык, ученик имеет возможность тут же проверить себя и в соответствующей реальной деятельности. При этом подчеркивается не самоценный, а прикладной характер компьютерной игры.

Вероятно, мы далеко не исчерпали возможный перечень важнейших функций компьютерной игры. Но все же предложенный перечень уже дает возможность приблизиться к психологической классификации игр и разработке принципов их применения в психолого-педагогической практике.

До сего времени в литературе чаще встречаются классификации игр скорее по внешним признакам — по их технологическому устройству, по способу обмена информации с игроком. В логических играх (типа «Раскрытие преступления», «Мастер-майнд» и др.) осуществляется текстовой диалог между компьютером и игроком. В позиционных играх («Шахматы», «Шашки», «Реверси», «Го», «Морской бой» и др.) используется графическая репрезентация некоторых сменяющих друг друга по определенным правилам статических, дискретных позиций. В динамических (или двигательных) играх игрок управляет с помощью клавиатуры (или спецустройств ввода — джойстика, тумблеров, «мышки») движением графических объектов на экране. В отличие от позиционных игр динамичная игровая среда не ждет хода игрока — изменяется в реальном режиме времени. В вероятностно-стратегических играх («Кости», «Однорукий бандит») диалог может быть оформлен и как текстовой, и как графический, но главное — расчет вероятностей возможных исходов. К вероятностным стратегическим играм приближаются экономические, для которых, как известно, разработан прежде всего аппарат математической теории игр. В экономических играх эксплицирована классификация интересов игроков: антагонистические (конфликтные), бескоалиционные, кооперативные.

Эта традиционная классификация сегодня должна быть дополнена классификацией игр в терминах развития перцептивных, психомоторных, мнемических, интеллектуальных, коммуникативных и эмоционально-волевых качеств человека.

В специальной детализации нуждается интеллектуальная сторона игр. Нужны классификации, дифференцирующие дедуктивное (формально-логическое) и

креативное (эвристическое) мышление. Второму, по-видимому, принадлежит более важная роль в становлении психотехнической игровой культуры. Некоторыми авторами основным моментом компьютерных курсов игрового обучения объявляется вербализация (осознание и словесное формулирование) различий между возможными стратегиями в игре. Это осознание множества вариантов — этап становления особой активности по осознанному выбору стратегии, что является необходимым компонентом психотехнического действия (собственная исходная стратегия осознается из рефлексивной позиции внешнего наблюдателя — как лишь одна из возможных потенциальных стратегий).

Нужно детализировать возможные классы игр, требующих развития кооперации, сотрудничества в ходе совместного обучения. Как показали недавние исследования, кооперативная мотивация приводит к более высоким результатам компьютеризованного обучения, чем конкурентная мотивация. В коллективах, увлеченных совместной деятельностью учения, достигается необходимый эффект ролевой инверсии: продвинутые ученики автоматически занимают позицию учителя по отношению к отставшим. А задача коммуникации (обучения другого) требует более осознанного выделения наиболее существенных признаков материала.

Разработка психологической классификации компьютерных игр находится на начальной стадии. Требуются согласованные усилия компьютерно-грамотных психологов — целого коллектива, располагающего возможностью проверять свои гипотезы в экспериментальной практике.

Хотя вопрос о вооружении играми школьного психолога-консультанта некоторым представляется преждевременным, а некоторым уже перезревшим, тем не менее реальная эффективность психолого-педагогической службы зависит не от проницательности отдельных исполнителей, а от разработки дифференцированной системы научно обоснованных рекомендаций по индивидуализированному применению игр. Это необходимо и практикам-консультантам, и педагогам-предметникам, и самим учащимся.

Польза от таких рекомендаций выразилась хотя бы уже в том, что они активизировали бы работу ума по осознанию психологической направленности игр, т. е. фактически способствовали бы становлению культуры психотехнического применения компьютерных игр. С этой точки зрения, научные рекомендации — лучшая альтернатива бесконтрольному стихийному распространению заразительной инфантильной игромании, к сожалению неизбежной детской болезни первых лет компьютерного всеобуча.

Н. ЮНЕРМАН
Новосибирск

Информатика в школьном кружке

Сейчас во всех школах нашей страны читается курс «Основы информатики и вычислительной техники». Помимо уроков занятия с учащимися IX—X классов ведутся на факультативах, в кружках школьных и внешкольных учреждений.

Как правило, программы таких факультативов и кружков рассчитаны на старшеклассников, но многие преподаватели занимаются информатикой и с ребятами среднего школьного возраста. Учащиеся IV—V классов легко осваивают основные понятия информатики, операторы несложного языка программирования и приобретают первые навыки в написании программ. Иногда педагог сталкивается с проблемой: какие задачи предложить ребятам этого возраста, чтобы заинтересовать их. Вместе с тем задачи должны быть доступны для решения учащимся IV—V классов.

В Новосибирском Академгородке работа со школьниками младших классов ведется около 10 лет. Сначала — в кружках школы юных программистов. С 1983 г. вычислительный центр Сибирского отделения АН СССР и Министерство просвещения СССР проводят эксперимент в школе № 166 Академгородка. Обучение ведется по экспериментальным программам и методикам.

Проводя занятия в кружке, педагог может предложить учащимся IV—V классов упражнения по мере изучения школьниками основных предписаний языка программирования. Такие упражнения могут быть использованы в работе учащимися независимо от того, какой язык программирования они изучают. В приведенных примерах занятия со школьниками велись на персональных ЭВМ «Агат», оснащенных системой «Школьница».

Педагоги, проводя занятия со школьниками, отмечают, что основные предписания языка программирования изучаются в следующей последовательности:

вывод;
присваивание;
процедуры без параметров;
ввод;
циклические предписания;
условные предписания;
процедуры с параметрами.

Прежде чем перейти к описанию задач, сделаем несколько общих замечаний к их использованию.

Рассматривается система так называемых расширяемых задач, т. е. новые понятия и предписания, вводится на примере уже решенной задачи путем ее усложнения.

Предполагается, что программы для решения задач допускают многократный запуск с различными начальными данными, после чего полученные результаты должны быть проанализированы.

При написании программ большое внимание уделяется выбору мнемоничных имен, комментариям, записи лесенкой, оформлению диалога.

После изучения темы «Предписания вывода и присваивания» можно предложить школьникам такие задачи:

1. Дана длина стороны квадрата. Определить его периметр и площадь. Проанализировать, как изменяются периметр и площадь в зависимости от длины стороны (сравнить рост линейной и квадратичной зависимостей).

2. Даны длины сторон прямоугольника. Определить его периметр и площадь. Зафиксировав длину одной из сторон и изменяя длину второй стороны, проанализировать изменение периметра и площади.

3. Даны два целых числа. Выполнить операцию «деление с остатком», т. е. определить частное и остаток. (Эта задача интересна при изучении тех языков программирования, где нет стандартных операций для нахождения частного и остатка при делении целых чисел.) Зафиксировав делитель и изменяя делимое, определить, какие значения может принимать остаток при заданном делителе. Сформулировать общее правило.

4. Из одинаковых книжных полок со-

ставлен шкаф. Известны высота каждой из них и общее число полок в шкафу. Определить его высоту. По известной высоте комнаты определить: шкаф из какого числа полок может в ней поместиться.

5. Известно, сколько рублей стоит обучение одного школьника в год и число учеников в каждом классе школы. Определить, сколько стоит обучение всех учащихся школы в год. Результат оформить в виде таблицы.

6. Дан радиус окружности. Определить ее длину. Проанализировать, как изменяется длина окружности в зависимости от радиуса.

После изучения тем «Процедуры без параметров» и «Предписания ввода» предложите учащимся оформить решения задач 1—6 в виде процедур, заменив начальные присваивания вводом исходных данных.

Дополнительно можно предложить учащимся следующие задачи.

7. Описать процедуру ПОКУПКА, которая запрашивает и вводит количество имеющихся денег, стоимость покупки и вычисляет величину сдачи. Рассмотреть два случая:

а) все величины даны в копейках;

б) величины могут быть даны как в копейках, так и в рублях и копейках.

8. Описать процедуру ПУТЬ, которая запрашивает и вводит скорость и время движения и определяет пройденное расстояние.

9. Описать процедуру УРОЖАЙ, которая запрашивает и вводит площадь поля и урожайность с 1 га и определяет размер урожая со всего поля.

10. Описать процедуру СМЕНА, которая запрашивает и вводит продолжительность смены рабочего и его производительность труда, определяет дневную выработку.

11. Описать процедуру ТЕЛЕГРАММА, которая запрашивает и вводит число слов в телеграмме и определяет ее стоимость.

12. Описать процедуру ПЕРЕГОВОРЫ, которая запрашивает и вводит продолжительность разговора по междугороднему телефону и тариф для данного города и определяет стоимость разговора.

13. Дата представляется тройкой чисел: день, месяц и год. Известны даты — текущая и дня рождения человека. Определить, сколько дней он прожил (считать, что в каждом месяце 30 дней, а год содержит 365 дней).

После изучения темы «Циклические предписания» можно опять вернуться к задачам 1—13 и рассмотреть ситуации, когда указанные действия необходимо выполнить много раз. При этом нужно разработать различные способы решения в зависимости от того, известно или нет заранее число повторений.

После изучения темы «Условные предписания» к задачам 1—13 следующие вопросы даются для двух ответов на них — положительного или отрицательного.

Имеется проволока заданной длины. Хватит ли ее для изготовления каркаса прямоугольника с данными сторонами?

Определить, делится ли нацело одно число на другое?

Число полок задано. Поместится ли составленный из них шкаф в комнате заданной высоты?

Хватит ли имеющихся денег на покупку?

Известно расстояние между городами. Можно ли доехать от одного до другого, если известны скорость движения и время?

Известна дневная норма рабочего. Выполнит ли он ее, если известна его производительность труда?

После изучения темы «Процедуры с параметрами: все рассмотренные задачи можно описать в виде процедур с параметрами, выбрав в их качестве исходные данные.

Для самостоятельного решения учащимся можно предложить следующие задачи.

14. При подаче телеграммы указывают ее вид (обычная или срочная) и тип (простая или поздравительная). Описать процедуру, которая запрашивает и вводит вид и тип телеграммы, количество слов в ней и определяет ее стоимость.

Указание: в срочной телеграмме стоимость одного слова 10 копеек, в обычной — 5 копеек. За поздравительную телеграмму берется дополнительная пла-

та в размере 10 копеек. За подачу любой телеграммы — сбор 20 копеек.

15. Описать процедуру, которая запрашивает и вводит число этажей в доме, квартир на этаже, номер квартиры и определяет этаж и номер подъезда данной квартиры, необходимость пользоваться лифтом (лифтом пользуются только в том случае, когда квартира находится выше второго этажа).

16. Запрограммировать игру «Отгадай число». Машина задумывает число из некоторого интервала, а человек должен его отгадать. При каждом ответе человека машина сообщает, больше или меньше задуманное число. В конце должно выдаваться сообщение о том, что число угадано, и указывается, с какой попытки. Если есть возможность, можно сделать «цветной» диалог: сообщения машины выдавать одним цветом, а ответы человека — другим.

17. Решить заданное уравнение (неравенство) с одной переменной в целых числах на заданном интервале.

Указание: использовать метод полного перебора.

18. Описать процедуру для вычисления $n!$. Проанализировать, как изменяется $n!$ с ростом n.

В следующих задачах решения линейных уравнений с двумя неизвестными в целых числах приводят к использованию конструкции «вложенные циклы». В задачах надо определить интервалы для каждой из переменных, для которых ищется решение. Все числовые данные можно заменить переменными, организовав ввод их начальных значений.

19. У мальчика было 37 копеек. Он купил тетради по 2 копейки за штуку и карандаши по 3 копейки за штуку. Сколько тетрадей и карандашей он мог купить одновременно на все деньги (указать возможные сочетания)?

20. У гусей и кроликов вместе 32 лапы. Сколько могло быть кроликов и гусей?

21. Груз перевозят в пяти- и трехтонных автомобилях. Сколькими машинами каждого типа можно перевезти 105 тонн груза?

22. Как трех- и двухкопеечными монетами можно набрать 30 копеек?

Рекомендуемая литература

Иоффе А. Ф. Массовые персональные ЭВМ серии «Агат» // Микропроцессорные средства и системы. 1984. № 1.

Звенигородский Г. А. и др. Программная система «Школьница» и ее реализация на персональных ЭВМ // Микропроцессорные средства и системы. 1984. № 1.

Звенигородский Г. А. Первые уроки программирования. М.: Наука, 1985. Серия «Библиотечка «Кванта».

Е. ПОЕЗД, А. ПОЕЗД

Кроссворд не для забавы

Кроссворды широко распространены практически во всем мире. Это один из наиболее популярных видов игр—головоломок. Составление кроссвордов (и в этом может убедиться каждый) — весьма трудоемкое дело, требующее известных навыков и значительного умственного напряжения. Удивительно, однако, что человеку удается составлять сложные кроссворды, ведь, с точки зрения математики, это кажется практически невозможным.

Пусть у нас есть достаточно большой словарь, и словами из него нужно заполнить некоторую кроссвордную сетку так, чтобы на пересечениях слов стояли одинаковые буквы. Формально эта проблема очень проста, является типичной переборной задачей, и стандартный подход к ее решению — использование метода перебора с возвратом. На практике такой прием не дает результата даже при применении быстродействующей ЭВМ. Дело в том, что число различных комбинаций, которые нужно проанализировать, оказывается ката-

строфически большим (комбинаторный взрыв). По разным оценкам, для достаточно сложного кроссворда это число равно 10^{30}—10^{120}. Для сравнения напомним, что число атомов в видимой Вселенной оценивается в 10^{70} (столько же примерно и всех возможных партий в шахматах).

Если предположить, что какой-то гипотетический суперкомпьютер способен каждую секунду анализировать 10 млрд. комбинаций (реальные возможности лучших ЭВМ по крайней мере на два порядка ниже), то на 10^{30} вариантов у него уйдет около 10^{20} с. Поэтому, чтобы в свежем номере журнала мы увидели кроссворд, над его изготовлением суперкомпьютер должен был бы начать работу примерно 10^{12} лет назад, а это на несколько порядков больше времени жизни Вселенной.

Многие, несмотря на такие удручающие оценки, пытались заставить ЭВМ составлять кроссворды. Пробовали, используя различные эвристические приемы, уменьшить число анализируемых комбинаций, но эти попытки к существенному успеху не привели и только доказали, что перебор с возвратом — это не тот алгоритм, который лежит в основе метода, используемого человеком при составлении кроссвордов.

Применялись и другие подходы к задаче составления кроссвордов на ЭВМ, но они тоже оказывались малоэффективными. Очень уж конструкции, изготовленные, скажем, с помощью основанного на методах прикладной теории вероятностей и математической лингвистики алгоритма Л. Мазлака, далеки от тех головоломок, которые мы привыкли видеть в газетах. Например, от кроссворда, публикуемого в нашем журнале.

Этот кроссворд составила ЭВМ с помощью программы, которая, по нашим сведениям, впервые позволяет компьютеру синтезировать достаточно сложные кроссворды. Известно, что в Турции составлен самый большой в мире кроссворд площадью 21 м^2. Это «достижение» зарегистрировано в книге рекордов Гиннеса.

Разработанный нами алгоритм позволяет превысить турецкий результат за несколько часов работы среднего компьютера (или за несколько минут на мощной ЭВМ).

В основе программы лежит принципиально новый метод, который является чисто алгоритмическим. На практике он, конечно, дополняется некоторыми эвристическими приемами, но они несущественны. Сам алгоритм имеет весьма общий характер и может быть применен при решении большого числа практически важных задач. По существу это не «алгоритм составления кроссвордов», а общий алгоритм синтеза систем, и широкоизвестная задача составления кроссвордов — просто удобная модель для эффектной иллюстрации его возможностей.

По основным параметрам (скорость, ресурсы памяти) работа алгоритма весьма близка к работе человека над составлением кроссвордов; есть основания полагать, что это не случайно. Алгоритм, видимо, несколько приоткрывает тайну того, как протекает работа мозга при решении некоторых задач, которые невозможно решить «в лоб», чисто логическим путем.

Возможности алгоритма исследуются, однако уже сейчас ясно, что его применение весьма перспективно во многих областях науки, техники, экономики. Системы автоматического проектирования, автоматизация интеллектуальной деятельности, экспертные системы, балансировка и состыковка планов, оптимизация графиков взаимных поставок, составление расписаний, выбор режимов работы энергетических систем, теория чисел и дискретная математика, где алгоритм дает принципиально новый подход ко многим задачам,— вот далеко не полный перечень областей, где он может быть эффективно использован.

В задаче составления кроссвордов алгоритм предоставляет очень широкие возможности для гибкого регулирования параметров синтезируемой головоломки. Единственное, где он сейчас «буксует» — при изготовлении очень сильносвязанных кроссвордов. В настоящее время, однако, создано (пока, правда, программно не опробованное) обобщение алгоритма, нацеленное на синтез сильносвязанных систем и позволяющее обой-

ти эту трудность. Можно, таким образом, утверждать, что проблема составления кроссвордов на ЭВМ решена полностью.

Имеющиеся программы для составления кроссвордов работают в пакетном режиме и действуют полностью автоматически. Используемый алгоритм допускает возможность гибкого диалогового управления процессом синтеза кроссвордов, но на примененных ЭВМ это по техническим причинам невозможно.

Одна из последних программ, улучшающих качество синтезируемых кроссвордов ориентируется на два критерия: использовать не более 5 слов, начинающихся на одну букву, и стремиться, чтобы как можно больше ключевых букв — букв, стоящих на пересечениях,— были редкими, согласными (при этом условии составить кроссворд значительно труднее, зато проще разгадать, ведь, например, редкая «Ч» несет гораздо больше информации и служит лучшей подсказкой, чем, скажем, «А»). Эти пункты программа успешно выполняет, и в этом вы убедитесь, когда разгадаете приведенный в номере один из сконструированных ею кроссвордов.

На его изготовление ушло около минуты на ЭВМ с быстродействием 10^6 операций/сек. При составлении использовался словарь объемом около 2 тыс. слов, который был сформирован из слов кроссвордов журнала «Смена» разных лет. Кроссвордная сетка была подготовлена вручную, хотя в программе есть блок для автоматического генерирования сеток. Общий объем программы — около 10 К, еще примерно 5К занимают различные вспомогательные массивы. Весь остальной объем оперативной памяти может быть использован под словарь. Слова хранятся в свернутом виде — буква занимает 5 бит, и на тысячу слов в среднем уходит 5К бит. Источники для толкования слов записаны на магнитной ленте и ищутся после того, как машина полностью составит головоломку.

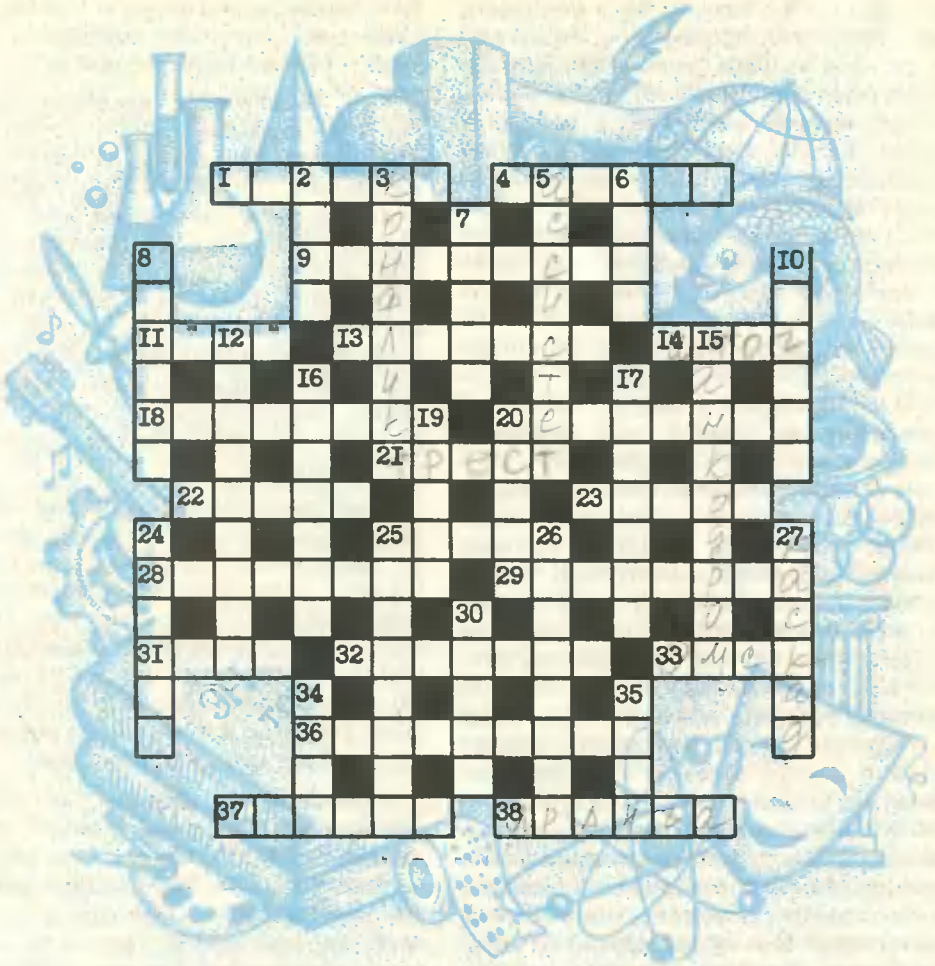
Приведенный кроссворд (среднее число пересечений, приходящихся на одно слово, равно 3,2) для программы весьма прост, и не характеризует ее максимальных возможностей. Со словарем в 2800 слов она легко составляет кроссворд с числом пересечений 4,2.

Как показывают исследования, для синтеза слабосвязанного простейшего кроссворда необходим словарь приблизительно в 700—800 слов. Если словарь слишком мал, программа дает обоснованный ответ, что составить кроссворд при заданных условиях невозможно. Для головоломок типа тех, что публикуются в газетах, достаточно 1200—1600, а для кроссвордов средней сложности — около 2000 слов. Случаи словарей, содержащих более 3000 единиц, не испытывались из-за трудности подготовки словаря.

Опыт программных реализаций алгоритма убеждает, что на его основе вполне возможна организация автоматического производства кроссвордов на любых языках, письменность которых допускает составление кроссвордов, причем изготавливаться они могут на любой вкус: общепопулярные, узкотематические, учебные, а в нашем журнале будут кроссворды и по информатике.

В заключение заметим, что возможность с помощью компьютера по специальному заказу получать в неограниченном количестве сильносвязанные высококачественные кроссворды открывает широкие перспективы для практического использования этих популярных головоломок в учебном процессе. Самостоятельная работа по «разгадыванию» учебных кроссвордов, подготовленных на базе программного материала, способна заметно оживить процесс обучения за счет внесения в него игровых моментов.

В ходе работы с кроссвордом учащийся сам выбирает стратегию разгадывания и, что очень важно, ему приходится постоянно возвращаться к попыткам вспомнить плохо усвоенные термины. При этом интересно, что задача как бы подстраивается под ученика — ведь постепенно подставляемые им в сетку слова дают подсказку для вспоминания более трудных. Особенно большую пользу «кроссвордная методика» может принести при освоении лексики иностранных языков.



По горизонтали:

1. Морская промысловая рыба. 4. Один из рулей самолета. 9. Смычковый инструмент. 11. Осенняя вспашка поля. 13. Город в Индии вблизи Дели. 14. Общая сумма, результат. 18. Река в Южной Америке. 20. Врожденная форма поведения. 21. Производственное объединение. 22. Торговое или промышленное предприятие. 23. Химический элемент, газ. 25. Военнослужащий специального рода войск. 28. Безосколочное стекло. 29. Медный духовой инструмент. 31. Освежающий напиток. 32. Алкалоид, применяемый в медицине для расширения зрачков. 33. Областной центр в Сибири. 36. Вид металлургической печи. 37. Документ об исключительном праве на изобретение. 38. Переносное жилище кочевых северных народностей.

По вертикали:

2. Единица измерения освещенности. 3. Столкновение, спор. 5. Сотрудник научного учреждения. 6. Стихотворение С. Есенина. 7. Пьеса М. Горького. 8. Река в Ереване, приток Аракса. 10. Спортивные соревнования на судах. 12. Советская детская писательница. 15. Место для испытания боевых машин. 16. Прямая, перпендикулярная касательной. 17. Пролив между Балканским и Апеннинским полуостровами. 19. Город во Франции, древняя столица исторической области Артуа. 20. Советский конструктор авиационных и ракетных двигателей. 24. Деталь машин в виде полого цилиндра. 25. Персонаж комедии Д. И. Фонвизина «Недоросль». 26. Устройство для охлаждения нагревающихся при работе радиотехнических элементов. 27. Остов сооружения. 30. Стихотворный размер. 34. Рыба отряда акулообразных. 35. Планета солнечной системы.

ЭВМ в народном хозяйстве

В. ЛЕВАДНЫЙ

Компьютеры в медицине

Общение с врачом всегда начинается с осмотра. Затем врач обязательно поинтересуется, чем вы болели раньше, или, пользуясь медицинским языком, выяснит анамнез. После мы получаем направления на анализы и в рентгенкабинет. Как видим, по существу, врач занимается тщательным сбором информации о больном. Данные, которые собирает врач, очень разнородны — здесь и цифровая информация (например, пульс, температура или величина артериального давления), и графическая (запись электрокардиограмм, например), и информация, представленная в виде особых изображений (рентгенограммы). Собрав нужные сведения, врач анализирует их для того, чтобы установить диагноз и выбрать нужную тактику лечения. При этом врач сравнивает полученные сведения с известными ему проявлениями тех или иных болезней, заглядывает в справочники и монографии — иными словами, работает с обобщенной мировой «медицинской памятью». В терминах информатики весь этот процесс можно назвать процессом распознавания.

Наконец диагноз поставлен, врач приступает к лечению. И опять он должен прибегнуть к помощи «медицинской памяти», ему нужно учесть, как действуют тысячи лекарств, насколько эффективны те или иные хирургические операции. Всю эту информацию он должен обработать и выбрать наилучший (а часто и единственный) способ лечения. И вот терапевтический курс или хирургическая операция проведены. Но процесс общения врача с больным на этом не заканчивается. Нужно установить, насколько успешными оказались медицинские воздействия на больного. Опять анализы, рентген, врач собирает новую информацию о больном; опять распознавание, принятие решения, коррективы в лечебной тактике, и вновь сбор данных,— процесс многократно повторяется.

Итак, формально процесс лечения выглядит следующим образом: сбор информации — обработка информации — распознавание (диагностика) — принятие решения — воздействие на больного — сбор информации — распознавание — ... . Казалось бы, раз этот процесс формализуем, его можно поручить ЭВМ. Наиболее горячие сторонники компьютеризации врачебной деятельности иногда даже утверждают, что когда-нибудь ЭВМ сможет заменить врача. С этим утверждением трудно согласиться. И не только потому, что из-за колоссальных объемов разнородной медицинской информации здесь могут возникнуть принципиальные ограничения, а скорее всего из-за гуманистического аспекта в деятельности врача. Как и всякий человек, врач смертен, ему доступны и понятны страдания больного. Принимая решение, он видит перед собой человека, он сопереживает больному. Возможно, именно это чувство и помогает найти врачу точ-

Автоматизированная система обеспечения решений врача помогает сотрудникам Института сердечно-сосудистой хирургии им. А. Н. Бакулева бороться с недугами.

ное и эффективное решение, которое чаще всего является результатом его интуиции, а не логики. Но врач всегда спешит, коварная болезнь не ждет. Поэтому в условиях жесткого дефицита времени чувство ответственности может парализовать интуицию врача. И тогда ему на помощь придет компьютер с его четкой, «бесчувственной» логикой.

Сейчас компьютеры только начали осваивать врачебную деятельность. На помощь врачу приходят другие специалисты. Физики, химики, биологии свои лучшие достижения и открытия внедряют в медицину. Электронная промышленность давно и успешно помогает медицине, современный врач использует в своей практике большое число сложнейших приборов. Теперь к ним присоединился и компьютер. Практически нет таких областей медицинской деятельности, в которых ЭВМ не могли бы принести пользы. С определенной долей условности можно выделить пять основных направлений, по которым идет их внедрение. (Здесь мы кратко охарактеризуем эти направления, а ниже некоторые из них опишем более подробно.)

Первые попытки использования ЭВМ в медицине связаны с автоматизацией простейших рутинных операций, выполняемых медицинским персоналом. Таким образом, первое (и надо отметить, наиболее успешное) направление — автоматизация медицинской деятельности.

Современная больница — это большое предприятие, со сложной организацией различных подразделений, с большими потоками медикаментов, продуктов и т. д. Если учесть, что эффективность работы этого сложного и дорогостоящего предприятия зависит не только от квалификации и численности медицинского персонала, от наличия аппаратуры и медикаментов, но и от качества и сроков обработки медицинской информации, то станет ясно, что без автоматизированной системы управления таким предприятием не обойтись. Реализовать такую систему можно, разумеется, толь-

ко с помощью ЭВМ. Медицинское учреждение, конечно, существенно отличается от, скажем, сталелитейного завода, однако в принципе АСУ больницы схожа с АСУ промышленного предприятия. Итак, вторая сфера приложения компьютеров в медицине — это АСУ.

Реакция человеческого организма на болезнь чрезвычайно индивидуальна. Это значит, что у одних больных инфаркт миокарда проявляется болью в области сердца, а у других болью в области живота, для одних наркоз совершенно безвреден, для других больных он может оказаться опасным. Поэтому так скрупулезно описывают врачи в своих публикациях проявления тех или иных болезней, а в каждой больнице еженедельно собираются врачи на конференции, чтобы рассказать коллегам о течении болезней своих пациентов. Создаваемая таким образом всеобщая «медицинская память» играет огромную роль в работе каждого врача. Одно из самых важных профессиональных качеств врача — память. Высококвалифицированный врач помнит огромное множество историй болезни, видов лекарств (сейчас только в Европе имеется около 40 тыс. наименований лекарств), учитывает их эффективность, совместимость, побочные действия и т. д. Банк данных, реализованный на небольшой ЭВМ, сможет выдать врачу исчерпывающую информацию как о лекарствах, так и о болезнях. Назовем эту третью сферу деятельности ЭВМ в медицине информационно-справочной.

Современная медицинская техника и методы клинической биохимии достигли такого уровня, что врач в кратчайшие сроки получает информацию о функционировании всех систем больного: аппаратура может выдавать ежесекундно значения более 400 параметров жизнедеятельности организма. Врач должен в короткое время воспринять, обработать эту информацию и принять решение. Если больной в тяжелом состоянии, врач фактически использует лишь небольшую часть полученной информации, а то и вообще махнет рукой на все эти сверкающие чудеса техники и, наклонившись к больному, по цвету губ, окраске ногтей или влажности кожи найдет единственное решение, которое и спасет больного. Обилие суперсовременных приборов, вместо того чтобы помочь врачу, мешает ему. Дорогостоящие приборы работают вхолостую. ЭВМ могут кардинальным образом изменить сложившуюся ситуацию. С помощью специальных программ компьютер быстро и эффективно обработает информацию, поступающую от приборов, и выдаст врачу несколько альтернативных решений, из которых он сможет выбрать наилучшее. В результате врач сможет выявить более тонкие проявления болезни, в более ранние сроки обнаруживать начинающиеся осложнения и предотвратить их. Итак, четвертое направление использования ЭВМ в медицине — это диагностика. Ниже мы несколько подробнее рассмотрим диагностическое направление использования ЭВМ в медицине.

Еще одна сфера действия ЭВМ в медицине — «работа в реальном масштабе времени». Так в информатике называют специальный режим ЭВМ, управляющий реальным физическим процессом.

Бывают ситуации, требующие мгновенной обработки информации. Например, при сложных хирургических операциях на сердце. Больной находится под глубоким наркозом, грудная полость вскрыта, сердце отключено — работает аппарат искусственного кровообращения. Операцию ведет хирургическая бригада из 10—12 человек, каждый занят своим делом. Но выдержит ли ослабленный организм все эти воздействия? Не нужно ли увеличить скорость кровотока? Уменьшить ли подачу наркоза? На помощь хирургу приходят приборы, которые информируют его о состоянии каждого органа больного. ЭВМ мгновенно обработает информацию о состоянии больного и каждую минуту будет сообщать: «Все в порядке, можно вести операцию дальше» или: «Внимание! Наркоз неадекватен, может наступить шок».

Другой пример работы ЭВМ в реальном масштабе времени — непрерывное слежение за состоянием больного после тяжелой операции. Это довольно ответственный этап в лечении, ни на одну секунду нельзя оставлять пациента без

внимания. В этом случае ЭВМ выполняет роль квалифицированной сиделки. Датчики непрерывно подают в память ЭВМ информацию о частоте пульса и кровяном давлении больного, замеряют глубину и частоту дыхания, анализируют газовый состав выдыхаемого воздуха, записывают ЭЭГ и ЭКГ. Компьютер всю эту информацию обрабатывает и в случае ухудшения состояния больного вызывает врача. Компьютер может и сам оказать некоторую помощь больному до прибытия врача, например включить аппарат искусственного дыхания, увеличить или уменьшить подачу кислорода.

Впервые ЭВМ появились в клиниках в начале 60-х г. За прошедшие два десятилетия произошло много перемен — появились новые, более совершенные компьютеры, были разработаны специальные алгоритмы и программы, ориентированные на медицинское использование. Вся эта область деятельности (иногда называемая медицинской кибернетикой) превратилась в самостоятельное, быстро развивающееся направление с широким диапазоном исследований. О некоторых результатах этих разработок мы расскажем подробнее.

Автоматизация медицинской деятельности

Обычно болезнь начинается очень незаметно, человек может годами нормально жить и работать, не подозревая, что болен. Только тонкие биохимические анализы смогли бы обнаружить начавшийся патологический процесс, и чем раньше проделать такие анализы, тем больше будет шансов на успешное лечение. Но как выявить таких потенциальных больных? Медицинская наука, в принципе, умеет это делать с помощью массовых профилактических осмотров населения. Однако в масштабе всей страны решение этой задачи наталкивается на огромные трудности. Для того чтобы тщательно обследовать каждого жителя нашей страны, необходимо затратить около миллиарда человеко-часов работы квалифицированного медицинского персонала. И это только для сбора информации. А сколько же потребуется времени на ее обработку? Проблема упрощается, если прибегнуть к помощи ЭВМ.

В настоящее время в Минздраве СССР заканчивается разработка типовой Автоматизированной системы профилактического осмотра населения (АСПОН), способной провести за одну рабочую смену тщательный осмотр 120 человек. АСПОН обеспечивает автоматический сбор медицинских данных, их обработку, выдачу итоговой документации. Так как АСПОН будет частью более крупной вычислительной сети, то она сможет легко обмениваться информацией как с другими АСПОН, так и с автоматизированными системами более высокого уровня. Важным преимуществом АСПОН является способность накапливать медицинскую информацию в больших объемах. В АСПОН будет накапливаться информация в течение многих лет, обработав эту информацию, можно будет выявить факторы риска, характерные для данной местности.

АСПОН состоит из нескольких подсистем. В подсистемах «Регистратура», «Антропометрия» и «Анамнез» накапливаются регистрационные данные о текущем состоянии организма, а также информация о прошлых болезнях обследуемого. В подсистемах «Кровообращение», «Дыхание», «Зрение», «Слух» собирается информация о конкретных органах и системах жизнедеятельности человека. Например, подсистема «Кровообращение» снимает и обрабатывает электрокардиограмму, частоту пульса, артериальное давление. В подсистеме «Флюорография» хранятся результаты рентгеновского обследования. Результаты лабораторных анализов хранятся и обрабатываются подсистемой «Лаборатория». После завершения обследования АСПОН обрабатывает собранную информацию и разделяет всех обследованных на несколько групп.

К группе практически здоровых относятся те из обследованных, все органы которых функционируют совершенно

В поликлинике Брежневского района столицы успешно работает автоматизированная система контроля и слежения за состоянием здоровья.

Отвечая на вопросы компьютера, пациент ведет диалог с ЭВМ, результатом которого станет вариант диагноза болезни.


5274 SS/SD
ИЗОТ
#9
СДЕЛАНО В БОЛГАРИИ

нормально. В группу риска попадают те обследованные, у которых выявляются некоторые отклонения в функционировании каких-либо органов, причем эти отклонения еще не являются собственно болезнью, но могут привести к болезни. Затем выявляются группы больных, каждая из которых соответствует той или иной болезни (сердечно-сосудистая, урологическая и т. д.). Если диагностика заболевания неоднозначна, то АСПОН выдает все возможные диагнозы, указав при этом вероятность каждого из них. В особую группу выделяются больные, требующие диспансерного наблюдения.

Как видим, в АСПОН компьютер обрабатывает огромный массив данных. Может показаться, что для этого требуются большие ЭВМ, которые есть только в специализированных вычислительных центрах. Но это не так. С помощью хорошо продуманного математического обеспечения такие системы можно реализовать на малых или даже на микро-ЭВМ. С 1983 г. в Латвийской ССР работает система, разработанная в рамках АСПОН и названная комплексной автоматизированной системой медицинского осмотра населения (КАСМОН). Вся система реализована на микро-ЭВМ «Искра-1256» и «Искра-226», сложная работа по распознаванию болезней выполняется на ДВК-2М. Работает система КАСМОН следующим образом: за несколько дней до обследования в системе формируются и распечатываются списки вызываемых на обследование жителей района (если это районная поликлиника) или сотрудников предприятия (если это здравпункт предприятия). По спискам высылаются приглашения, в которых указывается день и час, когда нужно явиться на обследование. Оно проводится группами по 20 человек, система КАСМОН пропускает восемь таких групп за одну рабочую смену.

В день обследования каждый пациент получает специальный жетон и «маршрут следования» по кабинетам. В кабинете анамнеза, например, каждый обследуемый располагается в кресле, перед которым стоит простое устройство с двумя кнопками «да» и «нет». Устройство также измеряет температуру. Обследуемый вставляет свой жетон в распознающий блок, система его идентифицирует, и он может приступить к работе с КАСМОН. Работать могут сразу все члены группы. На большом табло появляются вопросы (всего их 67), обследуемые отвечают на них при помощи двух своих кнопок. Собранные данные затем формируются в единый массив и обрабатываются на ДВК-2М.

Два года эксплуатации КАСМОН показали, что такие системы могут работать с большой эффективностью. КАСМОН — это новое и еще не очень привычное для врачей дело, поэтому большинство диагнозов, устанавливаемых КАСМОН, пока дублируется врачами с помощью традиционной методики. Однако данные ЭВМ более чем в 90 % случаев полностью совпадали с диагнозами врачей. В 1984 г. КАСМОН смогла обследовать 92,2 % населения республики. При этом из каждой 1000 обследованных у 123 были впервые установлены заболевания, о которых они и не подозревали. Всех выявленных больных КАСМОН разбивала на 16 классов (кардиологические больные, ревматологические, эндокринологические и т. д.). Для обслуживания системы требуется девять медсестер и один врач. Немаловажным достоинством системы КАСМОН оказалась ее экономическая эффективность.

Еще один важный аспект автоматизации врачебной деятельности связан с анализом и обработкой электрокардиограмм. Это одна из самых трудоемких медицинских работ, требующая больших затрат времени высококвалифицированных специалистов. Здесь также ЭВМ может оказать существенную помощь врачу. В медицинских центрах нашей страны работает автоматизированный комплекс аппаратуры, созданной в Горьком. Он предназначен для автоматической кодировки электрокардиограмм большого числа больных. Переведенные в машинные коды электрокардиограммы образуют в памяти ЭВМ специальный архив, который позволяет проводить оперативную диагностику различных заболеваний сердца. Такие компьютеризированные комплексы позволили существенно сократить время обработки

кардиологической информации и повысить достоверность диагноза заболевания.

Медицинская диагностика и прогнозирование заболеваний

Чтобы лучше понять возможности ЭВМ в диагностике заболеваний, рассмотрим два примера, описанные ведущим советским специалистом по машинной диагностике Е. В. Гублером в его книге «Вычислительные методы распознавания патологических процессов».

Вообразим следующую, довольно часто встречающуюся в медицинской практике ситуацию. В больницу поступил больной с острым заболеванием печени. Обследование и анализы позволили исключить целый ряд возможных диагнозов и оставить только два — это либо так называемая безжелтушная форма инфекционного гепатита, либо воспаление желчных ходов (холангит). Если справедлив первый диагноз, то больного нужно немедленно изолировать, второй диагноз этого не требует. Лечение этих заболеваний, разумеется, весьма различно. Однако сделать выбор между двумя диагнозами довольно сложно, состояние больного соответствует как одному, так и другому диагнозу. В такой ситуации помощь врачу может оказать компьютер. Известно, что пульс до 70 ударов в минуту встречается в четыре раза чаще при инфекционном гепатите, чем при холангите, а пульс свыше 76 ударов в минуту встречается в восемь раз чаще при холангите, чем при инфекционном гепатите (но пульс 71—75 ударов в минуту одинаково встречается при обоих заболеваниях). Далее, снижение числа лейкоцитов встречается в пять раз чаще при гепатите, чем при холангите. Допустим, что у нашего больного пульс ниже 70 ударов в минуту, а число лейкоцитов снижено до 4000 в одном $мм^3$. Следовательно, диагноз «инфекционный гепатит» у него в 20 раз вероятнее, чем диагноз «холангит». Это значит, что из 21 в 20 подобных случаях будет поставлен правильный диагноз. А в двадцать первом случае? Да, диагноз может быть ошибочным. Надо повысить достоверность диагноза, для этого нужно учесть дополнительные симптомы. Вероятность успеха увеличится, если подключить ЭВМ, в огромную память которой можно заранее ввести бесчисленные связи признаков и вероятностей заболеваний. Компьютер сможет учесть значительно большее число признаков, чем врач, что существенно повысит точность диагноза. В некоторых исключительных случаях ЭВМ сможет даже сделать то, что доступно лишь очень опытному врачу. Такие ситуации бывают, например, при тяжелых ожогах. Важнейшим параметром, определяющим тяжесть, прогноз и лечение таких ожогов, является площадь глубокого поражения кожи. Обычно эту величину определяют по внешнему виду обожженного участка, а для этого нужен опытный глаз специалиста по ожоговым болезням. Однако компьютер позволяет по таким показателям общей реакции организма, как частота пульса, артериальное давление, состав крови и т. д., отличить ожоги с площадью, превышающей 20 % тела, от менее тяжелых случаев.

Применение ЭВМ повышает точность диагноза. В нашей стране машинные методы диагностики давно применяются в клинической практике. Мы упомянем здесь лишь некоторые первые разработки, использование которых в медицинской практике уже прошло проверку временем. Одним из первых в нашей стране начал проводить работы в этом направлении известный Институт хирургии АМН СССР им. А. В. Вишневского. Вдохновителем работ был академик А. И. Берг, первые успехи этого института получили в свое время доброжелательную оценку Н. Винера, который посетил институт в 1962 г. В Институте хирургии уже десять лет на базе ЭВМ функционирует система дистанционной диагностики опасных заболеваний, объединенных термином «острый живот». Здесь создана система отбора оптимальной совокупности симптомов-признаков для дифференциальной диагностики и индивидуального прогноза исхода лечения онкологических больных.

Машинная диагностика болезней позволяет очень точно определять степень поражения такого важного органа, как сердце. Нарушения сердечной деятель-

ности, характерные для различных типов пороков сердца, могут внешне проявляться схожим образом. В то же время хирургу необходимо до операции точно установить степень поражения сердца. В этом ему может помочь разработанная в Институте сердечно-сосудистой хирургии им. А. Н. Бакулева информационная система, позволяющая диагностировать больных с врожденным пороком сердца, сопровождаемым переполнением малого круга кровообращения. Работа системы состоит из двух этапов. Вначале в памяти ЭВМ формируется так называемый информационный обучающий массив, в который вносятся данные комплексных обследований нескольких сот больных (обычно не менее 400 больных). В этом массиве выделяются больные с пороками сердца, сопровождаемые переполнением легочного кровотока. Для успешного функционирования системы необходимо иметь в массиве не менее 200 таких больных. Выделенная таким образом совокупность в свою очередь разбивается на подклассы с различной степенью порока сердца. После завершения формирования обучающего массива система переходит в режим нормального функционирования. В этом режиме медицинская информация о новом больном поступает в ЭВМ, где она сравнивается с данными «обучающего массива», обрабатывается и автоматически определяется та степень порока сердца, которой соответствует поступившая информация.

Очевидно, что подобным же образом могут быть сформированы информационно-диагностические системы, ориентированные на другие болезни. В том же Институте сердечно-сосудистых заболеваний хирургами разработан целый ряд систем, предназначенных для диагностики пороков сердца. Многолетняя эксплуатация этих систем показала, что диагноз, устанавливаемый с помощью ЭВМ, обладает высокой достоверностью. Например, система, предназначенная для диагностики приобретенных пороков сердца, использует информацию о 2000 больных и оперирует диагностической матрицей, рассчитанной на 200 признаков и 22 диагноза.

Интересно, что в некоторых случаях использование ЭВМ позволяет получить дополнительную информацию из результатов медицинского обследования больного. Например, в Каунасском медицинском институте разработана система, позволяющая использовать низкочастотные показатели электроэнцефалограмм для диагностики глубины наркоза.

Несколько в стороне от перечисленных выше направлений стоят работы, связанные с математическим моделированием тех или иных патологических процессов. Математическим моделированием в данном случае называется такой способ исследования и анализа болезней, когда закономерности патологических процессов в больных органах человека (или даже в целом организме) описываются на математическом языке, чаще всего с помощью дифференциальных уравнений различной степени сложности. Затем эти процессы исследуются и анализируются уже с помощью известных математических приемов. В этом смысле второй закон Ньютона является математической моделью процесса падения камня. При таком подходе математические методы используются прежде всего как «концентрированная логика», позволяющая в огромной массе различных, порой противоречивых данных выявить закономерности, определяющие динамику соответствующих процессов. В принципе математическое моделирование не требует обязательного использования ЭВМ, но в медицине, где объект исследования — организм человека, может быть описан, даже в самом грубом приближении, лишь с помощью огромного числа параметров, математические модели немыслимы без ЭВМ.

В национальном агентстве по аэронавтике (США) вот уже много лет разрабатывается глобальная имитационная математическая модель организма человека. Если такая модель будет создана, то на ней можно будет заранее «прогонять» любые хирургические операции, исследовать воздействие на больного тех или иных терапевтических курсов. Разумеется, такая модель должна учитывать (и предсказывать) тончайшие нюансы индивидуальной реакции

каждого больного. Представьте себе — к врачу приходит больной, врач его обследует и вводит результаты обследования в память ЭВМ, которая по специальной программе формирует математическую модель, описывающую этого конкретного больного. Затем врач вводит в ЭВМ различные варианты лечения больного, компьютер чисто теоретически создает течение всех физиологических процессов в организме больного в ответ на тот или иной курс лечения. Врач, проанализировав реакцию организма на каждый курс лечения, сможет выбрать наилучший.

Такая глобальная модель — это дело будущего. Сейчас больших успехов достигли работы, связанные с математическим моделированием отдельных патологических процессов, например воспалительных. Течение любого воспаления определяется множеством факторов. Например, размножение инфекционных агентов, их взаимодействие с имунной системой больного и т. д. Несмотря на сложность и разнородность этих факторов, удалось описать воспалительный процесс с помощью нелинейных дифференциальных уравнений параболического типа. Полученные уравнения оказались довольно сложными, поэтому анализ этих уравнений возможен только с помощью ЭВМ.

Одна из самых интересных математических моделей воспалительных процессов разработана в нашей стране для анализа течения туберкулеза легких. Важнейшим достоинством этой модели является то, что в ней учитывается также химиотерапевтическое воздействие на больного. Поскольку эффективность химиотерапии зависит от концентрации лекарственного препарата в зоне воспаления, математическая модель позволила разработать оптимальную стратегию химиотерапевтического лечения туберкулеза.

И в заключение упомянем об интересной работе, выполненной в Институте эпидемиологии и микробиологии им. Н. Ф. Гамалеи. Здесь построена и реализована на ЭВМ математическая модель распространения эпидемии гриппа на территории СССР. Огромная модель учитывает население 100 крупнейших городов и 28 экономических районов нашей страны, ежедневные перемещения пассажиров по железнодорожным, авиационным и автобусным линиям, динамику протекания болезни. Исходными данными для модели является информация о первичных заболеваниях гриппа в нашей стране. Модель выдает точный прогноз о степени и скорости распространения эпидемии в СССР. Первые же испытания модели во время эпидемий гриппа в 1972 и 1973 гг. дали прекрасный результат — система определила точный прогноз для 80 % городов нашей страны. По-видимому, нет нужды особо говорить о преимуществах такой модели — здесь и своевременная мобилизация медицинского персонала, и оптимальное распределение дефицитных лекарств, и подготовка дополнительного медицинского транспорта. Экономическая эффективность от внедрения этой математической модели в практику уже давно и многократно превысила все затраты на ее разработку.

ОТВЕТЫ НА КРОССВОРД (на с. 98)

По горизонтали:

1. Тюлька. 4. Элерон. 9. Контрабас. 11. Зябь. 13. Алигарх. 14. Итог. 18. Амазонка. 20. Инстинкт. 21. Трест. 22. Фирма. 23. Радон. 25. Сапер. 28. Триплекс. 29. Валторна. 31. Квас. 32. Атропин. 33. Омск. 36. Конвертер. 37. Патент. 38. Яранга.

По вертикали:

2. Люкс. 3. Конфликт. 5. Лаборант. 6. «Русь». 7. «Враги». 8. Раздан. 10. Регата. 12. Благинина. 15. Танкодром. 16. Нормаль. 17. Отранто. 19. Аррас. 20. Исаев. 24. Стакан. 25. Скотинин. 26. Радиатор. 27. Каркас. 30. Хорей. 34. Скат. 35. Уран.

Международная выставка

ЭЛЕКТРОНМАШ '86

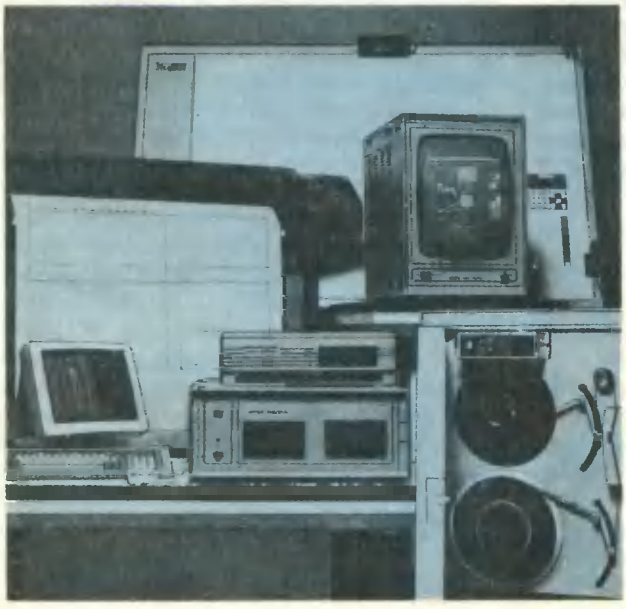
С продукцией многих зарубежных фирм познакомились посетители международной выставки «Электронмаш—86», которая проходила в Москве с 15 по 23 октября 1986 года.

Здесь была широко представлена разнообразная вычислительная техника. В экспозиции особое место занимали персональные компьютеры, используемые в обучении и профессиональной подготовке учащихся.

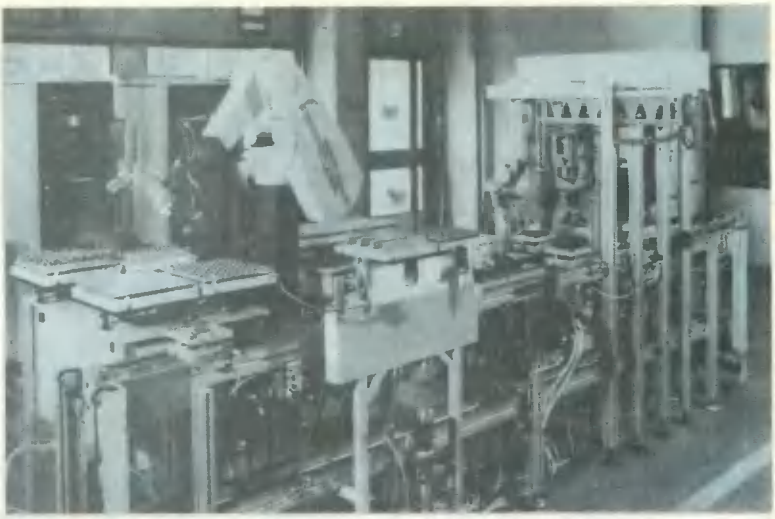
Выставка «Электронмаш—86» продемонстрировала успехи экономического сотрудничества СССР с другими странами в разработке и освоении производства вычислительной техники, ее техническом обслуживании.

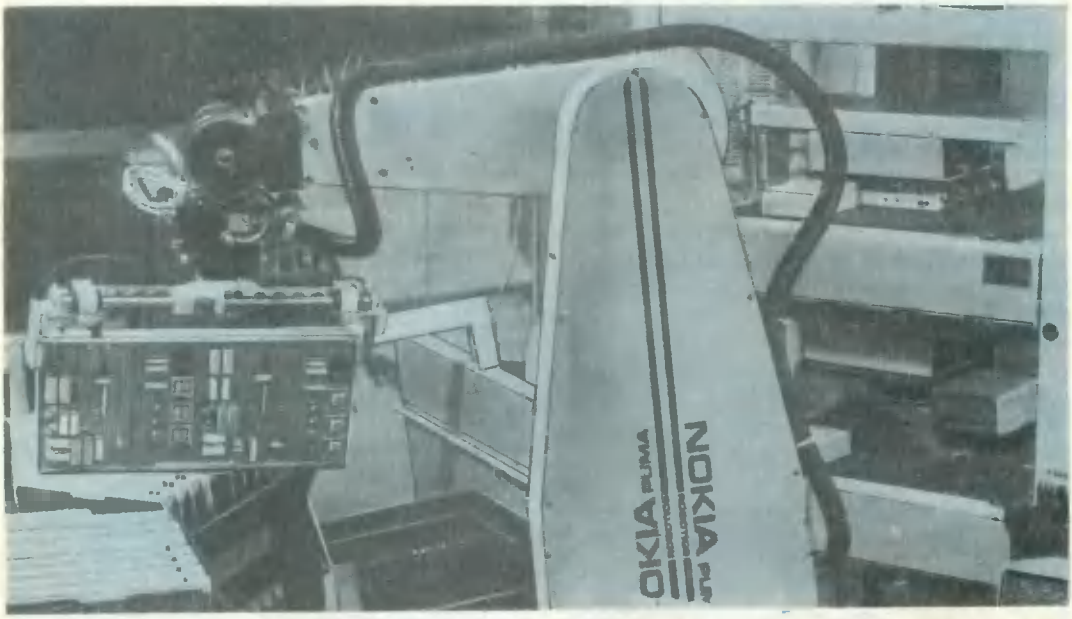
Система «АСК — МИКРО» (рис. 1) создана объединенными усилиями французских и советских специалистов. Она предназначена для контрольнопроверочных и наладочных работ, испытания разнообразной аппаратуры. В основе «АСК — МИКРО» французский компь-

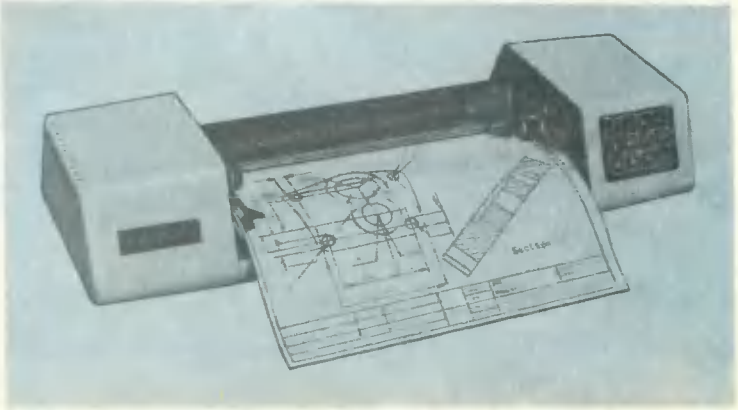
1



РЕПОРТАЖ НОМЕРА




NOKIA PUMA
NOKIA



ютер и советские приборы, рассчитанные на работу по программному продукту, созданному Минвузом СССР.

Первый договор о кооперированном производстве промышленных роботов был подписан советскими и финскими специалистами еще в 1982 году. С тех пор сотрудничество в этой области постоянно расширяется. В демонстрирующемся на выставке комплексе по зачистке, шлифовке, полировке деталей из металла и пластмассы микропроцессорная система управления (рис. 2) изготовлена в нашей стране, а робот «ПУМА» в Финляндии. О его возможностях уже рассказывалось в нашем журнале (ИНФО, 1986. № 1). Новая профессия этого робота связана с проверкой качества печатных плат, широко используемых в вычислительной технике (рис. 3).

С помощью графопостроителя КВЕСТ (рис. 4), представленного на выставочном стенде, учащиеся ПТУ могут изучать курс черчения. Одна из моделей такого графопостроителя свободно умещается на письменном столе. Подключенный к микро-ЭВМ графопостроитель в соответствии с учебным заданием готовит тот или иной чертеж детали или узла. Он может быть выполнен на бумаге, кальке, лавсановой пленке.

Вести диалог с ЭВМ учащимся поможет другой экспонат выставки — «мышь» МИКРОСОФТ — устройство управления с двумя клавишами, созданное западногерманскими конструкторами (рис. 5). Для быстрого освоения устройства МИКРОСОФТ созданы оргинальные обучающие программы.

5

Как отмечают советские и иностранные специалисты, международная выставка «Электронмаш—86» стала представительным смотром современной вычислительной техники и технологии ее производства, оказала положительное влияние на развитие международной торговли.

Посетители выставки познакомились с разнообразной вычислительной техникой.

Желающих поработать на персональных ЭВМ — экспонатах выставки было более чем достаточно.

Н. ЩЕТКИНА

Проводим эксперимент

Курс «Основы информатики и вычислительной техники» в IX классе предполагает обучение в безмашинном варианте. По инициативе кафедры алгебры и геометрии Петропавловского педагогического института для школ Северо-Казахстанской области предложен эксперимент по изучению параллельно с алгоритмическим языком одного из языков программирования с целью обработки программ, составленных учащимися на ЭВМ.

Язык для параллельного изучения должен удовлетворять следующим требованиям: 1) иметь высокий логический уровень; 2) быть доступным для начинающих изучать программирование; 3) иметь трансляторы на ЭВМ вычислительного центра.

Если по первому пункту требований выбор достаточно велик, то по второму и третьему число приемлемых языков резко сужается. Известно несколько языков для начинающих изучать программирование: Бейсик, Робик и др. Однако трансляторов этих языков на вычислительных центрах области нет.

Наиболее подходящим языком оказался Паскаль. Он обладает рядом преимуществ: по структуре и лексике близок к алгоритмическому, для него есть транслятор в ВЦ областного сельхозуправления.

Поэтому во время курсовой подготовки учителей к ведению курса «Основы информатики и вычислительной техники» были даны основы языка Паскаль. Учителя имели возможность обработать написанные ими программы на вычислительном центре. Они получили методические рекомендации по календарному планированию с учетом изучения двух языков, по оформлению кабинетов (таблицы ключевых слов, примеры алгоритмов и блок-схем и др.), знакомились с методикой ведения курса «Основы информатики и вычислительной техники», с параллельным изучением алгоритмического языка и языка Паскаль.

В целом было разработано 24 урока по информатике с параллельным изучением языков (сейчас готовятся к публикации).

Школы г. Петропавловска связываются непосредственно с вычислительным центром и передают программы для обработки самостоятельно. Областные школы присылают программы в пединститут, где их редактирует специальная группа студентов, входящая в кружок программистов. Протоколы обработанных программ с указанием ошибок, если таковые имелись, высылаются обратно для самоанализа. Колоды перфокарт оставались в ВЦ, и во время проведения экскурсий школьники могли сами посмотреть этапы обработки своих программ.

В эксперимент включилось около 43 % школ области. Из этого числа 33 % школ регулярно обрабатывают программы

в ВЦ. С 6 января по 14 мая на обработку школьных программ израсходовано 19,4 ч. К сожалению, бóльшая часть времени ушла на обработку программ городских школ — около 17 ч.

Наш опыт одобрен Новосибирским институтом информатики и вычислительной техники АПН СССР, вице-президентом АПН СССР академиком Ю. К. Бабанским. Всеми методическими службами будет продолжена работа по дальнейшему вовлечению учителей в эксперимент.

В качестве иллюстрации урока с параллельным изучением языка Паскаль и алгоритмического языка можно предложить разработку урока по теме «Алгоритмы и программы для работы с табличными величинами» (продолжение, урок второй по этой теме) учителя школы № 43 Л. А. Пенчуковой.

Цель и задачи урока: продолжить формирование навыков построения алгоритмов и программ для решения задач по обработке информации, собранной в таблице; развить культуру оформления программ.

Ход урока. Опрос четырех учащихся по карточкам у доски:

1) написать алгоритм «поиск»;
2) написать программу «поиск»;
3) написать алгоритм «счет» для определения количества элементов таблицы, удовлетворяющих какому-либо условию;
4) написать программу «счет».

Самостоятельная работа классу (пока отвечающие готовятся у доски): условие записано на доске, а задание каждый ученик получает на карточке.

Задание: исправить ошибки в записи алгоритма и программы пастой контрастного цвета. Записать условные задачи по составленному алгоритму.

Выполнение подобных заданий прививает ученикам навыки редактирования и культуры оформления программ, является необходимым условием для правильной обработки программы на ЭВМ.

После сдачи карточек учащиеся проверяют записи, сделанные отвечающими на доске.

Дополнительные задания и вопросы отвечающим у доски:

1) Записать команду вызова вспомогательного алгоритма.
2) Записать оператор вызова процедуры.
3) Для чего нужны вспомогательные алгоритмы?
4) Записать оператор READ для введения прямоугольной таблицы M[1...N, 1...K].

Изучение нового материала (тезисы).

1. Постановка задачи упорядочения: процесс нахождения нужного элемента таблицы будет происходить быстрее, если таблица будет упорядочена, т. е. если есть список фамилий, то удобнее расположить их по алфавиту, если числа, то в порядке возрастания или убывания, и т. п.

2. Остановиться на таблице вещественных чисел A[K:N]. Чтобы ее упорядочить, нужно несколько раз выполнить поочередно две следующие операции: а) найти наименьший элемент, б) поменять его местами с первым из рассмотренных чисел.

3. Разбор алгоритма для нахождения наименьшего элемента таблицы. В целях экономии времени учитель отказывается от разбора блок-схемы и лишь рассматривает алгоритм с учащимися по учебному пособию (с. 59). Разработанный алгоритм устно исполняется для таблицы из чисел: 6,7 —1,5 3,8 18,6.

4. Составление учащимися программы на языке Паскаль делается на доске. Ученикам напоминают, что программу с алгоритма нужно списывать с учетом законов языка Паскаль.

5. Домашнее задание:

§ 6 до слов «Построим теперь...» (с. 59);

составить алгоритм и программу «мин. элемент» для прямоугольной таблицы K[1:N, 1:M];

исполнить алгоритм «мин.элемент» для таблицы

18,7 —4,1 —0,7 16,4 8,1

На этом уроке учитель развивал интерес к предмету самостоятельной работой учащихся и в домашнем задании, где им придется самостоятельно составлять алгоритм и программу для прямоугольной таблицы, (это усложняет задачу, разобранную в классе). Во время объяснения нового материала учитель активизировал работу учащихся, при-

влекая их к составлению программы, развивал их творческую активность, предлагая продумать варианты программы. Учащиеся хорошо знают служебные и ключевые слова, законы алгоритмического языка и языка Паскаль. Но иногда им трудно полностью составить алгоритм или программу. Учитель предлагает «списывать» программу с алгоритма.

В классе Л. А. Пенчуковой специальный ответственный отвозит на вычислительный центр программы, выполненные учащимися. Заводские шефы передали школе перфорирующие устройства, на которых ребята готовят программы к обработке на ЭВМ и доставляют подготовленные перфокарты на вычислительный центр. Кропотливая периодическая работа по обработке учебных программ позволила собрать внушительную библиотеку программ.

О. АПШАКОВ, Н. ПИСАРЕНКО

Информатика — педагогам

Двухнедельные летние курсы переподготовки учителей информатики и вычислительной техники на базе Орехово — Зуевского педагогического института были организованы МОИУУ (руководитель М. Г. Ратинский). Переподготовку прошли 140 учителей общеобразовательных школ, преподавателей ПТУ и техникумов Подмосковья в три потока.

Программа курсов, расчитанная на 72 ч, была достаточно обширной, ориентированной на сравнительно хорошо подготовленную аудиторию. Так, например, в разделе «Алгоритмический язык» давался материал и методические рекомендации только ко II части учебника «Основы информатики и вычислительной техники».

А контингент слушателей оказался довольно разнообразным. В каждом из трех потоков можно было четко выделить две группы: те, кто уже ранее бывал на курсах переподготовки учителей информатики и сам преподавал предмет, и те, кто впервые на подобных курсах, и им только предстоит начать преподавание.

Новичкам в первые два-три дня было нелегко, приходилось не только усваивать предлагаемый материал, но также дополнительно заниматься самостоятельно, изучая то, что уже хорошо было известно их более подготовленным товарищам. И нам, преподавателям, когда мы впервые столкнулись с такой аудиторией (на самом первом потоке), пришлось буквально на ходу перестраиваться, делать объяснения более подробными, чем это было запланировано ранее. У новичков не было еще той минимальной базы знаний, на которую мы рассчитывали, приходилось давать материал немного сверх программы. Но уже к концу первой недели занятий знания слушателей выровнялись, и у бывших новичков чувствовалась более полное, цельное понимание предмета.

Конечно, с теми, кто уже преподавал новый предмет, накопил свой методический материал, буквально выстрадал каждый урок, было легче работать. Они уже знали, что именно им понадобится в учебном процессе. Задавали конкретные вопросы, просили подробно разбираться в задачах, предлагаемых в учебнике, делились своими находками в методике ведения урока информатики. Например, были у нас учителя из Павлово-Посада, Шатуры, Загорска, у которых уже есть годичный опыт работы в компьютерном классе.

Мы, конечно, старались выполнить все просьбы слушателей, предлагали задачи, которые были бы им интересны. Там, где могли, давали методические рекомендации. Безусловно, учителям и преподавателям техникумов хотелось получить методические разработки уроков. Но этого мы предложить не могли, да такой задачи курсы перед собой и не ставили.

Нашей задачей было помочь слушателям сориентироваться в новом материале II части учебника «Основы информатики и вычислительной техники», разобраться в решении задач, изучить возможности работы с программируемым микрокалькулятором,— ведь многим предстоит безмашинный вариант преподавания, более подробно познакомить наших слушателей с языками программирования Фокал, Бейсик, обучить самостоятельной работе на персональных ЭВМ, а также пробудить творческий интерес в подходе к преподаванию нового предмета. Мы условно выделили в планируемом программой учебном материале три направления:

1) устройство ЭВМ — преподаватель Ю. В. Граблин;

2) программирование (программное обеспечение, языки программирования, прикладное программное обеспечение) — преподаватель Н. Г. Писаренко;

3) безмашинная информатика (алгоритмический язык и язык Рапира, который на наших ЭВМ не реализован) — преподаватель О. М. Апшаков.

За каждое из этих направлений отвечал один преподаватель, при этом учитывались его профессиональные интересы. Этот преподаватель читал лекции по соответствующим темам программы, а также проводил практические и лабораторные занятия по своей тематике. Поэтому у наших слушателей было как бы несколько учебных дисциплин.

Многие из них (особенно молодые слушатели курсов) были вполне согласны с такой формой проведения занятий, так как она повторяет обычную вузовскую систему преподнесения учебного материала. Однако некоторые слушатели высказывали пожелания, чтобы весь материал читался последовательно, логически законченными частями.

На занятиях по алгоритмическому языку была сделана попытка осветить некоторые вопросы, которые в школьном учебнике неизбежно остаются «за кадром». Например, проводилось сравнение алгоритмического языка с некоторыми современными языками программирования; был сделан анализ этого языка по стандартной схеме, т. е. описан его алфавит и выделены основные конструкции (константы, переменные, описания типов и т. д.); выяснялся уровень формализации алгоритмического языка в связи с возможностью его машинной реализации. На практических занятиях по алгоритмическому языку было разобрано подавляющее большинство задач из учебного пособия для X класса.

Несколько занятий было посвящено программируемым микрокалькуляторам. Следует отметить, что, на наш взгляд, эту тему не нужно было включать в программу летних курсов: во-первых, потому, что в отведенное на нее количество часов достаточно содержательный материал рассмотреть невозможно, во-вторых, так как в школах, как правило, нет программируемых микрокалькуляторов. В техникумах и некоторых ПТУ они используются, и для преподавателей этих учебных заведений имело бы смысл рассмотреть эту тему отдельно, причем более подробно.

Программа предполагала изучение языков программирования Бейсик, Фортран или любого другого реализованного на имеющихся ЭВМ. Мы изучали язык программирования Фокал, и в течение двух недель проводились практические занятия на этом языке. Чем был обусловлен выбор данного языка? Прежде всего тем, что на установленных у нас микро-ЭВМ «Электроника БК-0010» реализован именно этот язык, а также тем, что большинство подмосковных техникумов, ПТУ и общеобразовательных школ оснащаются в настоящее время компьютерными классами на базе этих ЭВМ. Следовательно, язык Фокал за короткое время (по крайней мере в Московской области) может в сфере общего и профессионально-технического образования получить широкое распространение.

Начав знакомство с программированием на языке Фокал с простых примеров на непосредственный счет, постепенно слушатели научились сами составлять небольшие программы, с успехом выполняли домашние задания. Среди задач, рассмотренных на практических занятиях по языку Фокал, были задачи

из раздела «Библиотека алгоритмов» учебника «Основы информатики и вычислительной техники». Структуру этих алгоритмов и запись их на алгоритмическом языке многие уже знали из опыта преподавания в девятых классах. Теперь же предлагалось записать эти алгоритмы на языке Фокал, реализовать их на конкретной ЭВМ и проанализировать особенности этой реализации. Мы старались обучить слушателей всем возможностям работы с машинной графикой. Они с большим интересом строили графики функций на координатной плоскости в выбранном масштабе. Преподаватели-физики предлагали задачи на движение тел с графической иллюстрацией. Появился творческий момент, когда слушатели находили варианты использования микро-ЭВМ не только на уроках информатики, но и на уроках физики или математики.

Мы всячески поддерживали этот интерес к программированию, предлагали посмотреть имеющиеся у нас прикладные, игровые программы, давали советы по составлению новых программ.

Конечно, за две недели даже при такой напряженной программе, все успеть было невозможно. Что-то не получилось, для чего-то не хватило времени. В конце каждого очередного потока слушателей мы проводили конференцию, чтобы выслушать мнение слушателей о курсах, их пожелания, замечания. Особенно активно проходила конференция после окончания занятий первого потока, на которой присутствовал методист МОИУУ Ю. Г. Галкин.

Было высказано пожелание больше учитывать интересы той категории слушателей, которые знакомятся с этим предметом впервые, увеличить количество практических занятий. Предлагалось рассматривать методику преподавания информатики, что сейчас, в виду отсутствия массового опыта обучения этому предмету, к сожалению, невозможно. (Этот вопрос обсуждался особенно бурно.)

Высказывалось предложение за счет увеличения срока обучения на летних курсах сделать их программу менее напряженной.

На конференции выступила слушатель курсов учитель математики и информатики средней школы № 26 г. Орехово-Зуево О. К. Павлова. Она познакомила слушателей с материалами научно-практической конференции, которая проходила в январе 1985 г. в МОИУУ, на тему «Преподавание основ информатики и ВТ» в IX классе в безмашинном варианте», рассказала о создании нового журнала «Информатика и образование».

И. СПЕВАК

В новом кабинете

В школах началось преподавание информатики. Перед учителями встает вопрос как,оборудовать новый кабинет. В первую очередь в кабинете информатики должны быть персональные ЭВМ или терминалы с выходом на одну машину, программируемые микрокалькуляторы.

Сеть электропитания должна быть выполнена по правилам техники безопасности. Необходимо предусмотреть централизованное отключение электроэнергии в кабинете и пусковой автомат на каждом рабочем месте. Розетки желательно сделать углубленные, выполнить их в одном корпусе с пусковым автоматом и закрыть металлической крышкой. При соблюдении всех этих условий столы могут располагаться как в обычном классе, иначе их необходимо ставить вдоль стен, делая розетки в малодоступных местах.

В нашем кабинете установлены ПЭВМ «Агат», которые обладают широкими возможностями для обучения программированию, а в дальнейшем и для производительного труда: составления программ и решения задач, предложенных нам шефствующим предприятием.

Кроме техники в кабинете в обязательном порядке должна быть научно-популярная, учебная и, конечно, занимательная литература. Желательно сделать и тематические выставки книг: «Алгоритмические язы-

ки», «ЭВМ в народном хозяйстве», «История развития вычислительной техники» и т. д.

Хорошо иметь в кабинете подборки вырезок из газет и журналов. Незаменимую помощь в поиске материалов из периодической печати могут оказать ученики, а учитель должен систематизировать все вырезки. Журналы «Наука и жизнь», «Радио», «Юный техник», «Техника — молодежи», печатающие материалы по информатике, должны быть представлены в кабинете.

Стены кабинета должны тоже нести инфомационную нагрузку. Не обойтись без стенда «Вычислительная техника». При изучении истории развития ЭВМ, в течение всего курса информатики полезно обращаться к стенду, где рассказано, что такое вычислительная техника, в таблице представлены основные поколения ЭВМ и их технические характеристики. Материал для этого стенда можно найти в «Энциклопедическом словаре юного математика» и в других источниках.

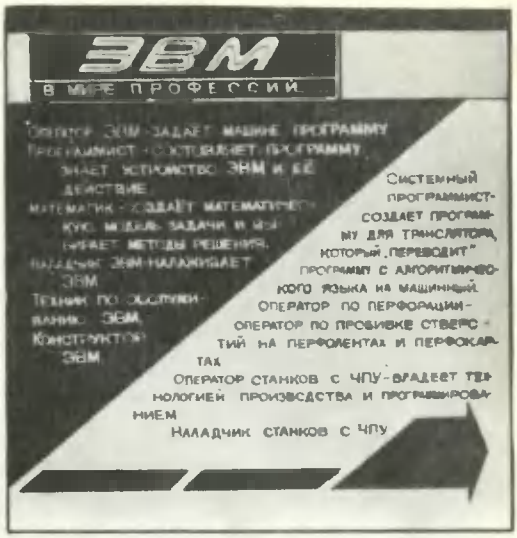
Второй стенд должен оказать помощь в изучении блок-схем и алгоритмического языка. Это стенд «Алгоритм», на котором приведено определение алгоритма, примеры алгоритмов, их свойства, блок-схемы: линейная, циклическая, разветвляющаяся, запись этих блоков на алгоритмическом языке, пояснения к условным обозначениям.

На следующем стенде можно привести служебные слова языка, который используется для работы в данной школе. У нас это «Служебные слова языка Бейсик».

Для целенаправленной профессиональной ориентации учащихся послужит стенд «ЭВМ в мире профессий» с кратким описанием профессии оператора-программиста, инженера по обслуживанию ЭВМ, оператора станков с числовым программным управлением, наладчиков ЭВМ и станков в ЧПУ, конструктора ЭВМ и т. д.

Полезно отобразить сферы применения ЭВМ в народном хозяйстве. Для этого послужит стенд «ЭВМ — всем, всем, всем!».

Этапы решения задач на ЭВМ представлены на стенде, материалы для которого мы взяли из журнала «Квант» № 9, 1985 г.



Такое внешнее оформление кабинета поможет лучшему восприятию материала и воспитанию учащихся. В нашем кабинете учащиеся работают на ЭВМ во время уроков, кружковых и факультативных занятий по информатике и программированию, освоили диалоговый и программный режимы, строят графики, рисунки, могут делать рисунки динамичными, менять цвета, добиваться инверсии, мерцания, составляют программы для решения уравнений методом проб и хорд, вычисления многочлена по схеме Горнера, для поиска простых и совершенных чисел, вычисления синуса, косинуса и других рядов, нахождения суммы и произведения членов некоторой последовательности. Есть у нас и программы-тесты для проверки знаний учащихся по арифметике и ряд игровых и шуточных программ. Начинаем составлять учебные программы и программы для проведения зачетов. Кабинет доступен для каждого, кто хочет научиться работать с ЭВМ. Для этого каждый день после уроков учащиеся, у которых готовы программы, могут получить у учителя консультацию и помощь по ее отладке.

Нам пишут

Н. СУКИЧЕВ
Учитель информатики
школы № 625, Москва

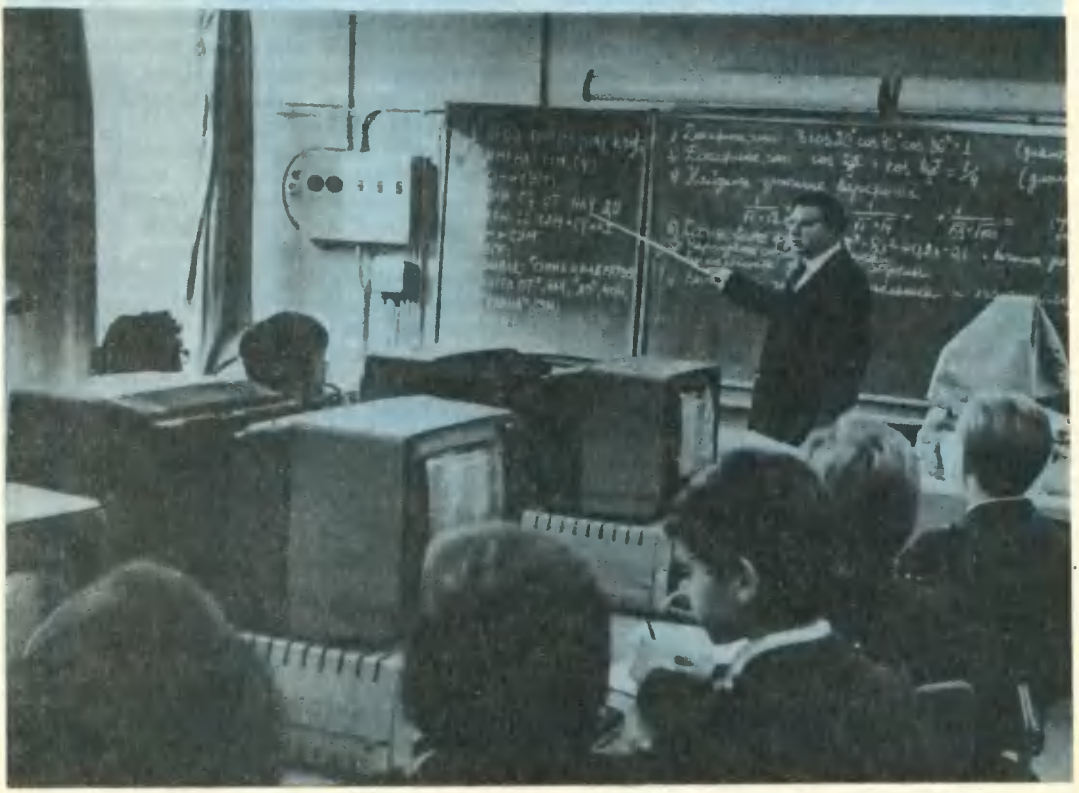
Наш кабинет

Кабинет информатики нашей школы в 1985/86 учебном году был оснащен персональными ЭВМ «Агат». Кроме обучения информатике он используется для ознакомления педагогов школ, является демонстрационным для делегаций учителей Москвы и других городов страны. Сотрудники Института физиологии детей и подростков АПН СССР работали у нас над вопросами охраны здоровья (в частности, зрения) школьников в дисплейном классе. Физиологи дали важные рекомендации по режиму работы учащихся, ограничив продолжительность занятий с дисплеем на уроке.

Преподают информатику в нашей школе сотрудники Вычислительного центра АН СССР. Они ведут на базе техники своего ВЦ обучение наших учащихся по специальности «оператор ЭВМ». Пока не был создан кабинет информатики, школьники составляли программы на языке Алгол и пользовались большой универсальной машиной ВЦ. Когда в учебный процесс ввели ЭВМ «Агат», учащиеся с радостью переключились на эту машину, отмечая, что работать с ней интересней; у «Агата» много игровых программ, есть цветная графика, используется удобный язык Бейсик.

Преподаватели сначала разрабатывали учебный материал самостоятельно, так как нужного учебника не существовало, однако в заданиях использовались и материалы учебного пособия по информатике для безмашинного обучения (автор А. П. Ершов). Педагогическая практика показала, что на уроках информатики удобно пользоваться карточками индивидуальных поурочных занятий. Желательно для учителей издать сборник дидактических заданий по информатике или публиковать карточки-задания в журнале «Информатика и образование».

В школах страны сейчас используются многие типы ПЭВМ, включая и зарубежную технику. Огромен интерес учащихся к этому многообразию, желание выяснить и сравнить возможности компьютеров, увидеть их достоинства. В этом большую помощь школьникам могут оказать диафильмы и кинофильмы, демонстрирующие ПЭВМ в сравнении, рассказывающие о работе с ними, показывающие работу программ и их структуру. Такие диафильмы особенно необходимы школам, не имеющим ЭВМ. Надо создавать диафильмы и по заданиям из школьной программы, сопровождая их показом компьютерных игр. Отметим, что на базе нашего кабинета и стендов ВДНХ СССР создан школьный



Ведет урок Н. А. Сукичев.

Если учеба идет отлично.

и настроение отличное.

диафильм «Архитектура и функционирование ЭВМ», который должен быть выпущен студией «Диафильм» в 1986 г.

В истекшем учебном году наши ученики изучали два языка программирования: Бейсик и Рапира. Обычно дети всегда отдавали предпочтение Бейсику, ссылаясь на большие удобства изменений и развития программы, на возможность связи с ассемблерными включениями и работы с Бейсик-программами от других машин.

Многочисленные игровые программы, которыми оснащена ПЭВМ «Агат», использовались для освоения компьютера и как образцы приемов выполнения программы. Школьники сами составили игровые программы с движением цветных объектов, тиром и подсчетом результатов. С интересом разбирали они имеющиеся на «Агате» звуковые программы, что позволило им самостоятельно создать несколько программ-мелодий, сопровождаемых красочными цветными рисунками и текстами.

С увлечением занимаются учащиеся серьезными игровыми программами на «Агате»: шахматами, шашками. Изучили ребята многие обучающие программы и игры на языках системы «Школьница», созданные учащимися Новосибирска, Латвии и Казахстана. Они освоили системные программы ПЭВМ «Агат»: графический и текстовый редактор, тесты, копирование, калькулятор, информационную систему, средства системы «Школьница». Дети, увлекающиеся математикой, создали программы построения и решения треугольников, матричных расчетов, логическую игру «Ним» и др.

Говоря об оснащении техникой, отметим, что в проекте кабинета надо было бы предусмотреть отдельное лабораторное помещение, где бы велись подготовительные работы, хранилось временно не используемое оборудование, носители, документация.

Компьютеры для школы должны обладать следующими качествами:

высокой надежностью при длительных нагрузках на каждый узел ПЭВМ;

наличием цветного дисплея, подходящего для детского зрения, имеющего антибликовую защиту, позволяющего совмещать графический и текстовый материал в одном кадре;

сетевой организацией связи учительской ЭВМ с машинами учащихся;

наличием у учительской ЭВМ большого демонстрационного монитора и принтера;

емкой оперативной и дисковой памятью и возможностью сохранения информации при кратковременном отключении ПЭВМ;

развитым математическим обеспечением, оснащением языками программирования, достаточно распространенными в сфере образования: Бейсик, Рапира, Лого, Паскаль;

проблемно - ориентированной модульностью.

В нашей школе были попытки использовать «Агат» для составления программ по различным школьным предметам. При этом оказалось, что было бы целесообразно, чтобы аппаратная часть ПЭВМ могла быть модифицируема в соответствии со спецификой предмета, т. е. имела модульную сборно-разборную конфигурацию из проблемно-ориентированных блоков. Например, для математики и химии, где много специальной символики,

Л. Эпштейн — автор игровой программы «Ним».

желательно иметь предметный алфавит клавиатуры и экрана. Для геометрии и географии нужна высокая графика монитора и графическое матобеспечение. Для физики нужен запас стеков: для измерительных применений ПЭВМ, для обмена ПЭВМ с внешними цепями цифровыми и аналоговыми сигналами, для управления моделями и др. На уроках литературы компьютер должен быть обеспечен принтером и текстовым редактором. Информационные системы для гуманитарных предметов требуют расширения оперативной памяти. То же можно сказать и о современных идеях создания универсальных обучающих систем, в которых учитель-предметник разрабатывал бы компьютерный урок, задавая ЭВМ входной сценарий.

В школах страны существует уже много кабинетов, оснащенных универсальными ПЭВМ, для которых создано и постепенно расширяется программное обеспечение. Для таких кабинетов переход на другие ПЭВМ и новое матобеспечение более сложен, чем постепенная модернизация, перевод на новые модели ЭВМ, архитектурно совместимые с предшествующими, при условии постоянного наличия у них развиваемого базового программного фонда. Целесообразно, чтобы модернизацию оборудования таких кабинетов в плановом порядке выполняла бы специальная организация.

А. ПАРАМОНОВ

Центр информатики в Бурятии

В 1985—1986 гг. в Бурятской АССР организован республиканский Центр информатики и вычислительной техники. В его создании активное участие принимал и обком ВЛКСМ.

Действует инициативная группа комсомольских работников, молодых ученых, учителей, студентов, учащихся ПТУ, старшеклассников.

Центр организован на базе Бурятского пединститута им. Банзарова, Восточно-Сибирского технологического института, ВЦ Бурятского филиала Сибирского отделения АН СССР.

В мае прошлого года в Центре проходила первая олимпиада студентов и учащихся по основам информатики и вычислительной техники.

Для учащихся общеобразовательных школ и ПТУ, изучающих основы информатики и вычислительной техники, подготовлена к печати брошюра «ЭВМ и информатика», в которой популярно рассказывается об истории развития ЭВМ, ее основных устройствах. Во время летних каникул на базе пединститута работал лагерь «Юный программист», где более

30 учащихся из разных школ республики три недели изучали основы информатики, познакомились с персональными ЭВМ («Ямаха», БЗ-34), программируемыми калькуляторами. Состоялись экскурсии в вычислительные центры города.

Во всех районах республики определены опорные школы по информатике и ВТ. За ними закреплены вычислительные центры предприятий и учреждений, создано 72 кабинета ВТ. В Октябрьском УПК Улан-Удэ введен новый профиль обучения — программист ЭВМ, в СПТУ-11 47 юношей и девушек обучаются по специальности оператор ЭВМ. Уже сейчас отмечается высокая заинтересованность предприятий и организаций в специалистах такого профиля. В Совмине и Минпросе республики получило поддержку предложение обкома ВЛКСМ о создании двух передвижных классов. Особый интерес представляют работы по организации компьютерно-телевизионных систем, проводимые в пединституте им. Д. Банзарова и ГПТУ-11. В городском Дворце пионеров и школьников выделено помещение для клуба юных программистов.

На факультете общественных профессий Восточно-Сибирского технологического института действует студенческий вычислительный центр. Комсомольцы разрабатывают программные средства для ЭВМ, организуют экскурсии школьников и учащихся ПТУ в ВЦ института.

Активное участие в деятельности Центра информатики и вычислительной техники принимают преподаватели информатики ГПТУ-11, учителя школы № 22 Улан-Удэ.

Но есть, конечно, в работе комитетов комсомола Бурятии по компьютерному всеобучу молодежи свои проблемы. Например, к пропагандистской и организационной работе Центра мало привлекаются комсомольцы-школьники, учащиеся ПТУ, молодые учителя. В республике ощущается нехватка наглядных и учебных пособий по курсу «Основы информатики и вычислительной техники». Существенную помощь в их изготовлении могли оказать бы комсомольцы. Однако комитеты ВЛКСМ эту работу практически не организуют.

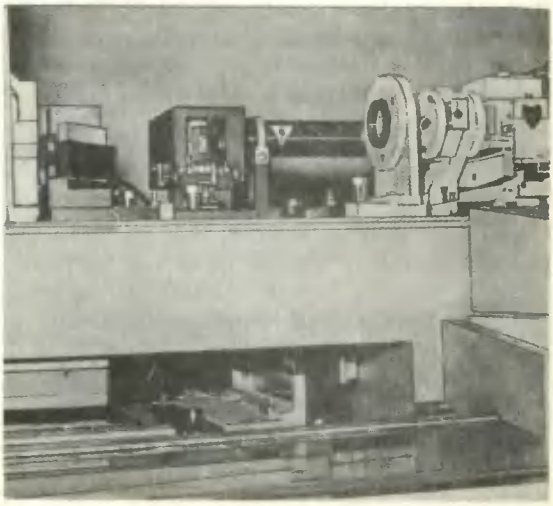
В молодежной газете мало публикуется материалов о компьютерном всеобуче. Не используется опыт Свердловской, Красноярской и других молодежных газет по ведению специальных рубрик, проведению заочных олимпиад, школ юных программистов. Мало привлекаются молодые специалисты к руководству кружковой и клубной работой.

ЭЛЕКТРОНМАШ '86

Для производства интегральных схем

Сверхбольшие интегральные схемы (СБИС) сейчас широко применяются в вычислительной технике. Одна такая схема состоит из нескольких сотен тысяч электронных элементов, хотя сама она свободно умещается на ладони. Производство интегральных схем — сложный и трудоемкий процесс, автоматизировать который поможет ЭВМ.

Советские ученые и инженеры разработали оригинальный комплекс, программное обеспечение которого согласуется с известной системой машинного проектирования «Кулон». Она обеспечивает большой выбор тестовых и диагностических программ, используемых при конструировании интегральных схем. В их производстве при изготовлении фотошаблонов применяется лазерная установка, управляемая компьютером (см. рис.). С ее помощью можно получать зеркальные и негативные рисунки фотошаблонов. В результате значительно увеличивается точность их изготовления, повышается качество интегральных схем.



Ю. ПЕРВИН, Л. ГОРОДНЯЯ

Юные программисты учатся

С 14 по 28 августа 1986 г. под Новосибирском в молодежном туристском центре «Сибиряк» собрались участники Всесоюзной летней школы юных программистов. Новосибирская летняя школа принимает гостей одиннадцатый раз. В школе 1986 г. приняли участие 225 человек из десяти союзных республик и зарубежные школьники из ЧССР и ГДР. Большинство присутствовавших — старшеклассники, хотя были среди участников ребята и младших классов.

Организаторы такой учебы — Министерство просвещения СССР, Президиум Сибирского отделения АН СССР и Научно-техническое общество им. А. С. Попова. В подготовке и проведении школы приняли участие такие научные учреждения, как Вычислительный центр СО АН СССР, НИИ информатики и вычислительной техники АПН СССР. Такой представительный состав организаторов смог обеспечить достаточно высокий уровень технической оснащенности школы. Здесь работали восемь отечественных ПЭВМ «Агат», 25 японских вычислительных машин «Ямаха» (в том числе один базовый кабинет), семь французских машин «Томпсон», объединенных в локальную сеть кабинета, и еще несколько одиночных машин. Каждый участник получил за время обучения в среднем более чем по 33 ч машинного времени — более трех часов ежедневно.

В учебном плане школы основное место занимали системы программирования на наиболее распространенных сейчас учебных машинах. Лекции и семинары по Рапире, Паскалю, Си, Бейсику, по ассемблерам «Ямахи» и «Агата», по операционным системам и редактирующим средствам, по графическим и музыкальным возможностям персональных ЭВМ были организованы для нескольких отдельных потоков учащихся. Распределение по потокам (и типам машин) в зависимости от уровня знаний учащихся и их пожеланий составило предмет серьезной оперативной работы учебного сектора.

Основная форма практической работы — индивидуальные и бригадные задания на проектирование пакетов прикладных программ. Вниманию участников был предложен богатый набор заданий — игровые задачи, учебные демонстрационные или контролирующие программы по различным школьным предметам, разнообразные задачи системного программирования, программные средства музыкальных применений учебных ЭВМ.

При разнообразии тематики разрабатываемых школьниками пакетов потребовалось обеспечить каждую бригаду компетентными консультантами. Группа консультантов из студентов Новосибирского университета и старшеклассников районной школы юных программистов новосибирского Академгородка

прошла предварительную подготовку в начале лета и в дни летней школы активно помогала как преподавателям, так и школьникам.

Важно было обеспечить повседневный контроль за выполнением побригадных заданий. Характерны в этом отношении требования постоянно действовавшей экспертной комиссии, отбиравшей лучшие из завершенных работ на итоговую конференцию научных работ школьников:

1. Четкая постановка задачи, в том числе схема диалога.
2. Краткое описание алгоритма.
3. Описание основных массивов, переменных, процедур и их параметров, входных и выходных данных.

Переменные должны иметь значащие имена.

Содержательные фрагменты программ должны иметь комментарии.

4. В процессе диалога программа должны пояснять, какого ответа она ожидает от пользователя.
5. Перед началом выполнения программы должны «представиться»: название и основные функции программы, авторы, дата, номер версии.

На рецензирование необходимо приносить листинг программы и заявку на машинное время. Без подписи рецензента заявка не рассматривается. Консультант бригады присутствует при рецензировании работы.

Распечатывать программы должны консультанты не во время, выделенное бригаде.

На две тематические секции итоговой конференции школы были представлены 38 индивидуальных и коллективных докладов, каждый из которых иллюстрировался результатами, выведенными на экраны персональных микро-ЭВМ.

Лекции и семинары по основным вопросам учебного плана школы, обзорные лекции ведущих ученых и специалистов из Москвы, Новосибирска, Ленинграда, работа над пакетами, две научные конференции школьников, а также широкая внешкольная программа с компьютерными играми, спортивными соревнованиями и вечерними кострами — все это в традициях новосибирских летних школ.

Была, однако, летом 1986 г. и одна новинка. Речь идет об экспериментальной группе «малышей» из шести мальчиков и девочек: одна — из II класса, трое — из III и двое — из IV. Им были предложены уроки информатики на языке программирования Лого. Три урока ежедневно в течение двух недель ребята проводили в специально для них выделенном кабинете — локальной сети французских персональных ЭВМ «Томпсон». Учебный план курса для этой экспериментальной группы был весьма интенсивным. Критерием отбора в группу был «нулевой» уровень умения программировать. Правда, все члены группы не раз играли до этого в компьютерные игры, но практически все основные понятия программирования они впервые узнали здесь. Диапазон, освоенный за две недели, оказался огромным: дети, начав с «черепашьей» графики и элементов музыкальной грамоты, освоили понятия процедуры и функции, процедуры с параметрами и цикла, логических значений и локализации переменных. Центральным пунктом методической работы в этой группе стало понятие рекурсии: показана эффективность этого механизма информатики, выяснены отношения рекурсии и циклических конструкций, рассмотрены разные виды рекурсий. Младшие школьники успешно работали с такими сложными понятиями, как структуры данных — тексты и списки, самостоятельно строили программы машинного исполнения цветомузыки и программы генерирования случайных «сочинений».

Результативность этого экспериментального курса — отнюдь не парадокс. Прежде всего задача эксперимента состояла не в закреплении практических навыков составления программ, а в усвоении фундаментальных понятий информатики, формирующих операционный стиль мышления учащихся. Эффективность уроков определялась тем, что школьники были избавлены от записей в тетрадях: к каждому уроку ученики получали дидактический и справочный материал так, что им оставалось только делать «записи» на экранах персо-

нальных ЭВМ. Не писал на доске и учитель: вместо доски использовался рулон бумаги, на котором до урока цветными фломастерами были выписаны все элементы урока. В ходе урока учитель должен был лишь разворачивать «доску»-рулон. Технические возможности локальной сети вместе с ее достаточной надежностью также можно считать существенным фактором, обеспечившим педагогическую эффективность уроков Лого. Впрочем, обращение школьников с машинами не ограничивалось уроками: пообедав, дети возвращались в свой класс и, хотя учебный план не предусматривал домашних заданий, подолгу засиживались за собственными программами.

По материалам этого эксперимента подготовлена рукопись монографии «Уроки информатики на Лого».

Не остались без внимания и руководители школьных делегаций. Впервые за всю историю Всесоюзных летних школ этот контингент оказался весьма компетентным. Это были школьные учителя информатики, вузовские преподаватели, руководители кружков программирования, организаторы клубов, станций, Домов и Дворцов пионеров, представители отделов и управлений информатики республиканских министерств просвещения. Для этой аудитории интенсивно работал методический семинар, на котором принципиально и живо учителя обсуждали опыт своего первого года обучения основам информатики в общеобразовательной школе и разнообразных формах внешкольной работы с учащимися в области информатики, проблемы методики обучения школьников и технического обслуживания кабинетов информатики. Руководители делегаций ЧССР и ГДР подробно рассказали о состоянии компьютеризации школьного образования в социалистических странах. Много вопросов накопилось у участников семинара ко встрече с академиком А. П. Ершовым — бессменным научным руководителем всех новосибирских летних школ юных программистов. На одном из заседаний семинара докладчиком была Л. Е. Самовольнова, представлявшая Управление информатики и вычислительной техники Министерства просвещения СССР. Стратегия и тактика управления компьютеризацией советской школы — тема ее выступления.

Впрочем, учительский семинар школы состоял не только из информационных выступлений и дискуссий за «круглым столом». В ходе работы семинара были подготовлены рекомендации Министерству просвещения в поддержку ежегодной Всесоюзной научно-методической конференции по школьной информатике. По просьбе Управления информатики Минпроса СССР семинар обсудил и разработал типовое положение о региональных летних школах юных программистов: начиная с лета 1987 г. в стране будут проводиться четыре такие школы, располагающиеся в известных центрах школьной информатики — Новосибирске, Вильнюсе, Симферополе и Переславле-Залесском. Решение о проведении региональных школ — новый этап развития внешкольных форм учебно-воспитательной работы с учащимися в области информатики.

Школа в Тбилиси

В соответствии с решением ЦК ВЛКСМ о проведении в 1986 г. всесоюзных школ молодых ученых и специалистов по актуальным проблемам науки, техники и производства и в целях повышения их творческой активности, совершенствования идейно-теоретической и профессиональной подготовки в Тбилисском международном молодежном центре «Золотое руно» с 16 по 23 октября проходила всесоюзная школа «Электронно-вычислительная техника в учебном процессе».

Во второй раз организуемая ЦК ВЛКСМ и Минпросом СССР совместно с АПН СССР школа собрала более 120 молодых ученых и преподавателей высших учебных заведений страны, учи-

телей, специалистов в области информатики и вычислительной техники — энтузиастов компьютерной грамотности.

Программа включала пленарные доклады, заседания по секциям, «круглый стол» с ведущими специалистами и учеными по школьной информатике, демонстрацию и тиражирование программного обеспечения ЭВМ.

«Внедрение электронно-вычислительной техники в учебный процесс требует прежде всего разработки педагогических основ использования ЭВМ в школе,— отметила, открывая школу, министр просвещения Грузинской ССР Н. Ш. Васадзе.— Советская педагогическая наука определила место компьютерного образования, которое призвано помочь учителю в совершенствовании учебно-воспитательной работы, в том, чтобы знания, навыки учащихся формировали идейно зрелого, грамотного и гармонично развитого молодого гражданина нашей страны».

В открытии школы принял участие и выступил с докладом о состоянии и перспективах использования электронно-вычислительной техники в учебных заведениях ведущих индустриально развитых стран мира заведующий отделом науки и учебных заведений ЦК Компартии Грузии Г. Г. Чоговадзе.

О проблемах компьютеризации образования, работе по консолидации усилий педагогов, программистов, психологов в разработке программ и учебников, учебных и наглядных пособий по информатике, необходимых программных средств для ЭВМ рассказал директор Научно-исследовательского института информатики и вычислительной техники АПН СССР И. М. Бобко.

Важные, требующие оперативного решения проблемы осветил директор НИИ СиМО АПН СССР В. М. Монахов. Он, в частности, отметил, что начатая кардинальная реформа школы затрагивает методические вопросы работы учителя, организации урока. Ожидается, что уже через пять лет мы вплотную столкнемся с широким использованием в учебных заведениях информационной технологии. Правомерна постановка вопроса: что даст она другим школьным предметам? На какой коэффициент полезного действия новой информационной технологии можем мы рассчитывать? Как повысится эффективность учебно-воспитательного процесса? Как пойдет нормализация учебной нагрузки?

На задачах комитетов комсомолов, советов молодых ученых и специалистов по привлечению молодежи к изучению и эффективному применению вычислительной техники остановился в своем выступлении инструктор ЦК ВЛКСМ А. И. Парамонов.

Широкий круг вопросов рассмотрен участниками школы во время секционных занятий. Состоялось более 30 выступлений и сообщений, было представлено 18 стендовых докладов.

Об опыте обучения преподавателей основам информатики и вычислительной техники, подготовке специалистов — программистов и операторов ЭВМ — рассказали научный сотрудник сектора школьной информатики ИПИ АН СССР Р. М. Полиенко и директор учебно-производственного центра Октябрьского района Москвы Т. П. Кравчук. Сравнительные характеристики отечественных персональных ЭВМ дал ведущий научный сотрудник Института электронных управляющих машин А. В. Гиглавый. Роли информатики в школьном обучении посвятил выступление заведующий кабинетом информатики Московского областного института усовершенствования учителей М. Г. Ратинский.

Особый интерес и активное обсуждение вызвали сообщения сотрудника кафедры информатики Ярославского педагогического института Е. В. Утлинского об опыте ведения машинного варианта обучения основам информатики и вычислительной техники, старшего научного сотрудника НИИ ШОТСО АПН СССР И. В. Роберт о педагогических требованиях к программному продукту, доцента кафедры кибернетики Московского института электронного машиностроения В. А. Каймина о компьютерной технологии обучения и технологии разработки обучающих программ.

О состоянии и перспективах совершенствования использования электронно-вычислительной техники в учебном процессе, оснащении школ техникой,

учебными и наглядными пособиями, педагогическими программными продуктами участникам школы рассказали сотрудники Управления информатики и электронно-вычислительной техники Минпроса СССР Т. В. Крюкова и Г. В. Баранова.

«Круглый стол», в проведении которого участвовали В. М. Монахов, А. В. Гиглавый, В. А. Каймин, Г. Г. Гварамия и другие, прошел в обстановке конструктивного, делового поиска путей решения проблем материально-технического, методического, кадрового обеспечения процесса компьютеризации школы.

Были рассмотрены психологические, нравственные аспекты использования электронно-вычислительной техники в учебном процессе. Участники школы обсудили содержание сборника научных трудов «Психолого-педагогические и психофизиологические проблемы компьютерного обучения», изданного АПН СССР совместно с МГУ им. М. В. Ломоносова.

Составной частью общего разговора за «круглым столом» стал разбор опыта работы комитетов комсомола Горьковской области по организации экспериментальных передвижных обучающих классов электронно-вычислительной техники, о котором рассказали преподаватель информатики школы № 174 г. Горького Л. Чиняева и член областного совета молодых ученых и специалистов В. Ю. Грачев.

Важное практическое значение имели демонстрация и обсуждение программного обеспечения для ЭВМ, представленного группами ученых и специалистов из Грузии, Украины, Москвы, Ярославля, Новосибирска, Благовещенска, Кишинева, Куйбышева и других мест.

Целенаправленный и конструктивный обмен мнениями об использовании электронно-вычислительной техники в учебном процессе позволил сформулировать некоторые рекомендации и предложения, выдержки из которых приводятся ниже.

В целях создания и совершенствования методики использования электронно-вычислительной техники в учебно-воспитательном процессе необходимо: осуществлять научно-методическое руководство работой коллективов разработчиков педагогических программных продуктов. Для обеспечения высокого уровня создаваемых программ считать обязательным участие в работе коллективов опытных педагогов, квалифицированных программистов, психологов, специалистов других отраслей. Решить вопрос о гарантировании их авторского права на соответствующие разработки. Особое внимание уделить программной поддержке курса «Основы информатики и вычислительной техники»;

считать целесообразным организацию работы творческих групп разработчиков педагогических программных продуктов на базе научных учреждений и учреждений, ведущих подготовку и повышение квалификации учителей;

головному институту АПН СССР по проблеме создания в системе народного образования единого фонда алгоритмов и программ (НИИ ИВТ) обеспечить работу по систематизации, оценке, поддержке и распространению программного обеспечения по курсу «Основы информатики и вычислительной техники» для всех типов ЭВМ, поступающих в учебные заведения по линии Минпроса СССР; осуществлять эту работу через соответствующие подразделения на местах, четко дифференцировав их функции и задачи;

провести работу по унификации инструментальных программных средств, определенных ТЗ на КУВТ «Корвет», «УК-НЦ» и другие типы ПЭВМ, поступающих на оснащение школ в переходный период; учитывать при этом существующую письменность народов СССР;

отработать механизм контроля и влияния на идеологическое и нравственное содержание учебных и игровых программ, разрабатываемых в нашей стране и приобретаемых за рубежом.

В целях создания условий, обеспечивающих необходимый уровень знаний, умений и навыков по основам информатики и вычислительной техники, следует:

обеспечивать первоочередную поставку стандартизованной электронно-вы-

числительной техники (с постоянным наращиванием и обновлением) в учебные заведения и организации, занимающиеся подготовкой и повышением квалификации учителей и инженерно-педагогических работников; не допускать поставки в систему народного образования некомплектной, неработающей и морально устаревшей техники; ускорить разработку типового договора на ввод в эксплуатацию и обслуживание вычислительной техники ремонтными предприятиями;

рассмотреть вопрос о единой номенклатуре информационных носителей, используемых для тиражирования программных средств КУВТ, их своевременной поставке в учебные заведения;

с учетом реальной возможности поставок техники поддержать обучение основам информатики и вычислительной техники в межшкольных кабинетах вычислительной техники (МКВТ); ускорить доработку и утверждение положения о МКВТ;

принять действенные меры к оснащению техникой молодежных компьютерных объединений (кружки, клубы, центры и т. п.), разработке рекомендаций по организации их деятельности.

Участники школы обратились с просьбой к Минпросу СССР об организации на Центральном телевидении постоянно действующей передачи в помощь учителям, методистам, преподавателям и специалистам, занимающимся проблемой компьютеризации образования.

Единодушно поддержано мнение о том, что такая школа должна быть постоянно действующей и проводиться в регионах страны, имеющих передовой опыт в обеспечении компьютерной грамотности подрастающего поколения.

Как писать в журнал «Информатика и образование»

Уважаемые читатели!

Если Вы хотите стать авторами нашего журнала, поделиться опытом методики преподавания курса «Основы информатики и вычислительной техники», организации занятий, внеклассной работы, технического творчества школьников в кружках электронного конструирования, вам необходимо помнить, что ваша статья, обзор, информация должны быть отпечатаны на пишущей машинке через два интервала на стандартных листах писчей бумаги в двух экземплярах. На последней странице рукописи надо указать фамилию, имя, отчество автора, домашний адрес, место работы, должность, количество детей в семье.

Когда рукопись готова и отпечатана, внимательно прочтите текст, уточните источники цитат, названия предприятий, учреждений, географические названия. Особое внимание нужно уделить проверке терминов, сокращений, названий, употребляемых в технической литературе. Все они должны соответствовать общепринятой терминологии, обозначенной в ГОСТах.

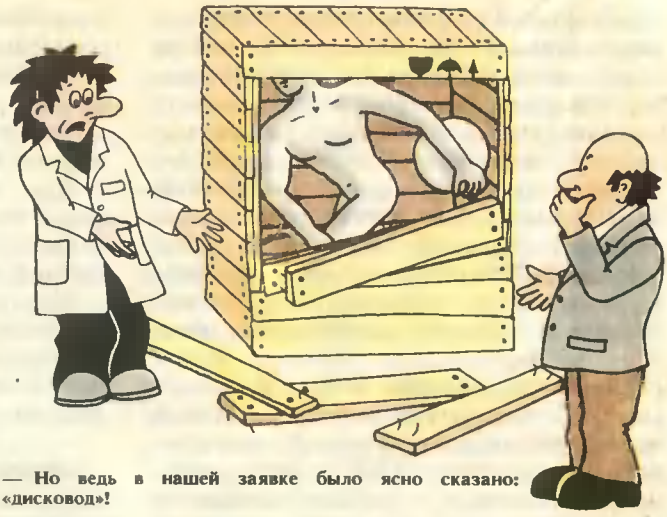
Эскизы, чертежи, схемы, прилагаемые к рукописи, должны быть выполнены в соответствии с действующими ГОСТами. Схемы и эскизы лучше нарисовать обычной шариковой ручкой на листах плотной бумаги.

Фотоотпечатки делаются на глянцевой контрастной бумаге размером не меньше 9×12 см. Фотографии должны быть без повреждений, изломов, царапин. Пояснительные надписи на них выполняются на обороте только карандашом.

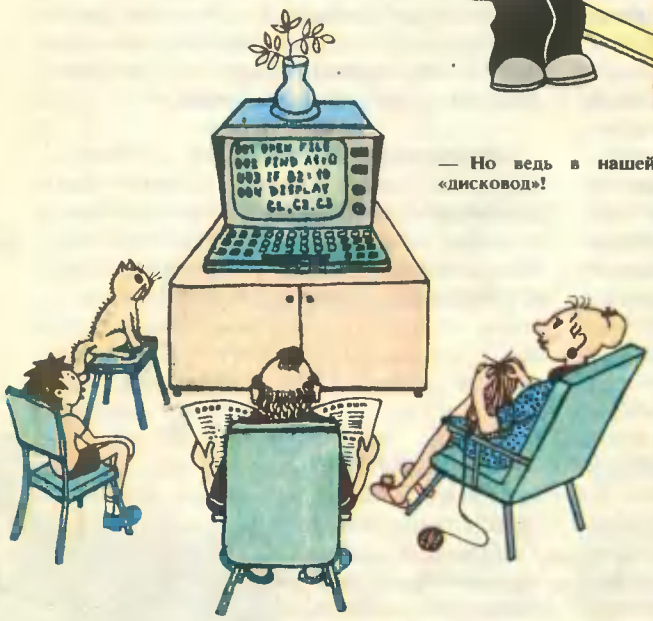
Готовую рукопись и иллюстрированный материал запечатайте в конверт и вышлите в редакцию. Наш адрес указан в журнале.

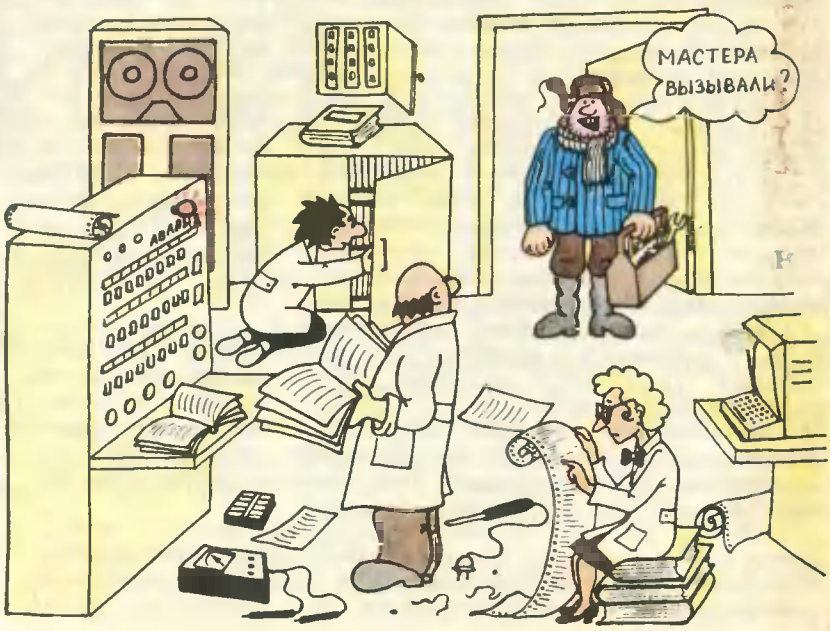
Ждем от вас, дорогие читатели, интересных материалов и предложений!

Весёлый урок



— Но ведь в нашей заявке было ясно сказано: «дисковод»!





На внеклассных занятиях учащиеся с увлечением работают с персональными ЭВМ, применяя их графические возможности для составления несложных программ, каждая из которых использует рисунки, выполненные компьютером. Они в основе компьютерных игр, развивающих у ребят навыки работы с ЭВМ, способствующих освоению курса основ информатики и вычислительной техники.

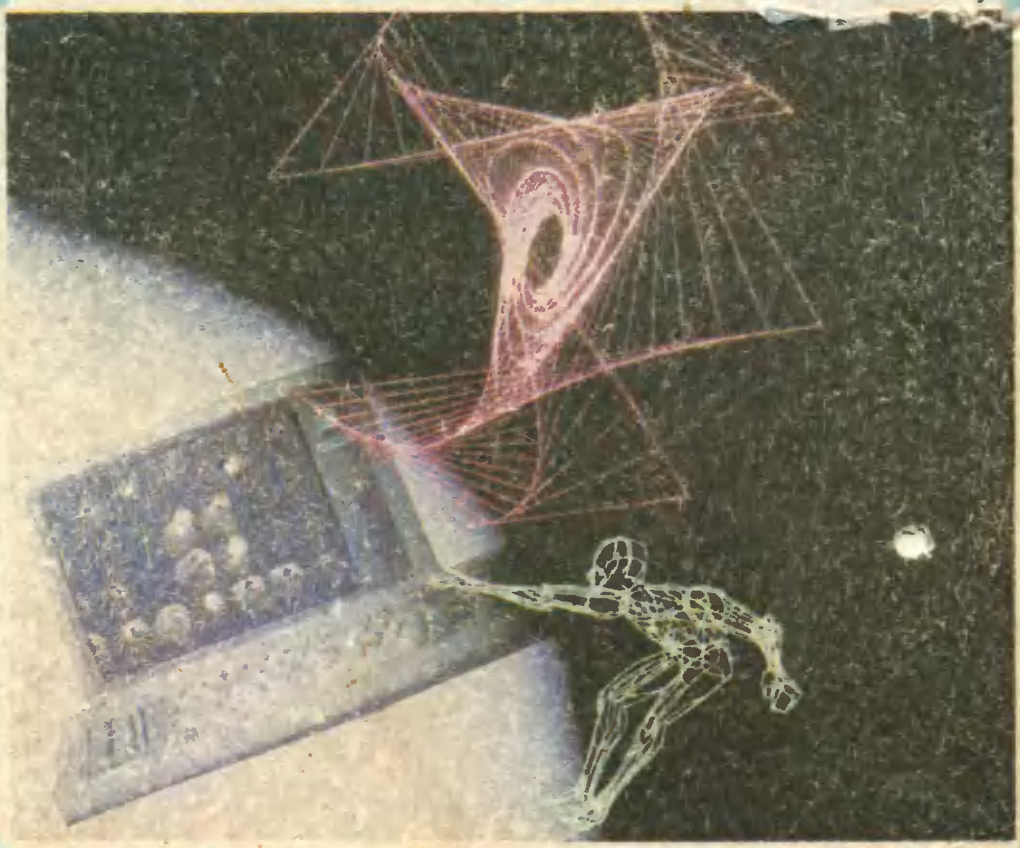
Цена 60 коп.

70473

ИНФО

1 '87

ИНФОРМАТИКА
И ОБРАЗОВАНИЕ



Информатика и образование. 1987. № 1