ISSN 0234—0453

ИНФО

ИНФОРМАТИКА И ОБРАЗОВАНИЕ

6 1987

Позвольте представиться, я «Агат».
Вы, конечно, слышали обо мне — как-никак
первый советский серийный
персональный компьютер.
Но, может быть, еще не видели?
Так вот мой самый лучший портрет

ночью,

шли,
ся мечтам
ошенной

рафии из

ческого
меня
нают,
тсюда я
льшой

обложки.)

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

ИНФ ИНФОРМАТИКА И ОБРАЗОВАНИЕ

6 '87
ноябрь — декабрь

Содержание

Обложка *Э. Бажилина*
В оформлении номера принимали участие *Э. Бажилин, А. Пономарев, С. Расторгуев*

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

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

Почтовый адрес: 107847, Москва, Лефортовский пер., 8
Телефон редакции: 249-97-77

Сдано в набор 25.09.87. Подписано в печать 04.10.87. А07437. Формат 70×100¹/₁₆. Печать офсетная. Усл. печ. л. 10,40. Уч.-изд. л. 15,11. Усл. кр.-отт. 42,88. Тираж 91 020 экз. Заказ 2765 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 области.

Общие вопросы

А. ЕРШОВ
Академик

Школьная информатика в СССР: от грамотности к культуре

Современное общество переживает беспрецедентное событие: впервые в истории народного образования предмет, который менее 50 лет назад даже не угадывался, в течение жизни поколения становится одной из ведущих наук, суммой профессий, затрагивающих десятки миллионов человек, наконец — элементом общечеловеческой культуры, вырабатываемым средним образованием. Для специалистов в области информатики быть даже только свидетелем этого процесса — уже незабываемый жизненный опыт. Автор позволяет себе надеяться, что осознание этого эмоционального подъема сделает читателя более снисходительным к черточкам субъективизма и персонализации, которые можно усмотреть в настоящей статье. Ее цель — дать очерк краткого, но решающего прошлого школьной информатики в СССР, ее весьма динамичного настоящего и будущего, подпадающего под магическое притяжение грядущей даты перехода в третье тысячелетие нашего летоисчисления. Школьная информатика сталкивается с множеством проблем. Еще больше их впереди. Однако прежде чем заниматься каждой из них в отдельности, желательно иметь общий взгляд на школьную информатику в целом.

Прошлое

Предпосылки. Начало школьной информатики относится к периоду, когда сначала с помощью систем коллективного пользования (разделения времени), а затем с помощью микро-ЭВМ и ПЭВМ удалось разместить информационно-вычислительные мощности и средства визуализации на парте учащегося. Предпосылкой реального начала стали наблюдения психологов, подчеркнувшие роль активной деятельности в учебном процессе. Другой предпосылкой оказалось ставшее возможным благодаря микро-ЭВМ и растровым дисплеям привнесение на стол учащегося существенно более развитых «моделей мира», допускающих интерактивное взаимодействие. И, может быть, самой главной, важной предпосылкой стала практическая потребность в появлении «компьютерно грамотного» поколения молодых людей в связи с массовым внедрением вычислительной техники в виде ПЭВМ и встроенных микропроцессоров. Все это привело к лавинообразному развитию событий в школьной информатике, наблюдаемому в 80-е гг.

Одно из мест, где сформировалась интегрированная концепция школьной информатики, подкрепленная определенным практическим опытом,— Новосибирский научный центр СО АН СССР. Здесь сложился неформальный коллектив научных сотрудников ряда академических институтов (прежде всего Вычислительного центра) и Новосибирского университета, педвузовских и школьных преподавателей, которые провели в 1976—1984 гг. достаточно широкую программу экспериментов, исследований и разработок. Компонентами этой работы были:

районная школа юных программистов;

летние всесоюзные школы юных программистов;

заочная школа юных программистов (совместно с журналом «Квант»);

общественная школьная лаборатория производственного программирования;

факультативы по программированию в старших классах;

трудовое обучение программированию в межшкольном учебном комбинате;

разработка учебных языков программирования «Робик» и «Рапира», системы базового программного обеспечения «Школьница»;

разработка экспериментальных обучающих программ по разным предметам;

педвузовские спецкурсы и курсовые проекты.

Автор далек от мысли представить дело так, что концепция современной школьной информатики сложилась исключительно в Новосибирске. Все эти годы активно работали группы специалистов в Тарту, Таллине, Вильнюсе, Минске, Киеве, Тбилиси, Харькове, Казани, Свердловске, Кургане, Барнауле, Красноярске — не говоря уже о Москве, Ленинграде и ряде других мест. Новосибирский коллектив осуществил, можно сказать, некоторую интегративную миссию, а также высказал ряд догадок и положений, которые впоследствии оказались подходящей моделью для государственной программы.

Вся работа по школьной информатике проходила на фоне постоянного осмысления тех потенциальных перемен и потребностей, которые привнесла современная вычислительная техника. Основными этапами этого анализа стали: реакция на первое знакомство с микро-ЭВМ [1], исследование особенностей программного обеспечения и применения ПЭВМ [2] и автоматизация работы служащих как сфера массового применения вычислительной техники [3].

Сам термин «школьная информатика» появился в литературе в 1979 г. в работе [4]. Ее важной особенностью был системный подход к проблеме и учет социальной роли вычислительной техники. Уместно напомнить одну выдержку из этой работы:

«Таким образом, для значительного большинства людей, использующих ЭВМ в своей работе или в быту, знакомство с конкретными языками и системами необходимо не в большей степени, чем телефонному абоненту нужны технические подробности телефонного аппарата и АТС.

В то же время для эффективного использования возможностей вычислительной техники при любой форме взаимодействия с ней необходимо владеть определенным стилем мышления, определенными навыками умственных действий, наиболее явно обнаруживаемыми сегодня у профессиональных программистов.

Наиболее существенные из этих навыков:

умение планировать структуру действий, необходимых для достижения заданной цели при помощи фиксированного набора средств;

умение строить информационные структуры для описания объектов и систем;

умение организовать поиск информации, необходимой для решения поставленной задачи;

умение правильно, четко и однозначно сформулировать мысль в понятной собеседнику форме и правильно понять текстовое сообщение;

привычка своевременно обращаться к ЭВМ при решении задачи из любой области;

наличие минимальных технических навыков взаимодействия с ЭВМ».

Положение об алгоритмических и программистских навыках как о фундаментальном компоненте человеческой деятельности в современном обществе получило свое дальнейшее развитие в работе [5], где была пущена в оборот метафора о программировании как второй грамотности человека. Являясь скорее риторической фигурой, нежели научным понятием, эта метафора, однако, содействовала внедрению идеи «компьютерной грамотности» в общественное сознание.

По мере накопления опыта, а также под воздействием общемирового развития возникла потребность в разработке организационных форм компьютеризации школы на уровне национальной программы. Одной из первых попыток подобного рода стала работа [6], доложенная в начале 1984 г. на международной конференции по информатике в Дрездене. Ключевыми ее положениями были идеи о кабинете школьной информатики (вычислительный кабинет) и о способах его использования в учебном процессе, а также выдвижение альтернативы Бейсику в виде учебно-производственного языка высокого уровня с родноязычной лексикой.

В 1982 г. Минпрос СССР принял решение о начале постепенного продвижения микро-ЭВМ в систему образования в качестве как технического средства обучения, так и области познания. Министерство радиопромышленности СССР начало разработку школьного компьютера «Агат», разработка СО АН СССР системы базового программного обеспечения «Школьница» была включена в государственный план. Ряд общих положений о школьной информатике был включен в готовящиеся документы по школьной реформе.

Школьная информатика в канун перестройки. Основополагающие документы по школьной реформе [7, 8] торжественно провозгласили компьютеризацию одним из важных направлений развития школы. Предстояло наполнить это стратегическое положение конкретным содержанием.

Одна из первых «формул» компьютерной грамотности была предложена автором в сентябре 1984 г. в его докладе совещанию руководителей народного образования СССР, посвященному школьной реформе [20]:

«Владение алгоритмической нотацией в объеме, достаточном для выражения вычислительных планов и программ для скалярных, векторных, структурных и текстовых величин, содержащих циклы, ветвления и процедуры. Понимание связи алгоритмической и общематематической нотаций. Умение составить программу решения задачи на материале общеобразовательных дисциплин и предметов трудового обучения. Знакомство с принципами устройства ЭВМ, владение начальными навыками обращения со школьной ЭВМ, умение применить школьную ЭВМ к решению практических задач на материале общеобразовательных дисциплин и предметов трудового обучения. Представление о возрастающей роли ЭВМ в жизни общества, знание конкретных примеров применения ЭВМ».

В первые годы компьютеризации школы (1982—1985) провозглашалась «инфильтративная» стратегия, подразумевавшая постепенность и выборочность внедрения идей и методов информатики в школу. Этот подход, однако, стал приходить в противоречие с нарастающими темпами общемирового процесса. Кроме того, обнаружилось, что эта стратегия не в состоянии преодолеть инерцию и неадекватный уровень общественного сознания, а также вскрытый впоследствии «механизм торможения» развития общества в целом. Стала ясной необходимость существенно более энергичных усилий, перехода от «инфильтративной» стратегии к фронтальному наступлению. И соответствующее решение было принято [9].

Надо честно признаться, что в недрах научных и педагогических кругов такое крутое решение не могло бы быть принято. Это было политическое решение, и как таковое оно оказалось своевременным и правильным. В результате все пришло в движение и за последовавшие шесть месяцев было сделано больше, чем за предшествующие шесть лет, причем разгон не ослабевает до сего дня. Вот далеко не полный перечень событий этого периода «бури и натиска»:

разработка начальной программы 68-часового курса «Основы информатики и вычислительной техники» для IX—X классов (ниже для краткости курс будет называться «информатика») [10];

написание I части пробного учебного пособия по информатике (IX класс) и его массовое издание [11];

написание и издание методического пособия к I части пробного учебного пособия [12];

курсовая подготовка 100 тыс. учителей по информатике в мае — июне 1985 и 1986 гг.;

организация специальности «Информатика и вычислительная техника» в педвузах страны;

разработка технических требований к массовым школьным компьютерам;

разработка программной поддержки начального курса информатики (Е-практикум) [13];

поставка вычислительных кабинетов на базе ПЭВМ «Агат» [14], КУВТ-86 [15];

закупка японских ПЭВМ «Ямаха» и их распределение по школам страны;

завершение разработки и сдача во всесоюзный фонд алгоритмов и программ системы программного обеспечения «Школьница» [16];

организация управлений и отделов информатики и вычислительной техники в министерствах просвещения СССР и союзных республик, областных отделах народного образования, институтах усовершенствования учителей, в системе профессионального и среднего специального образования;

организация и развертывание в Новосибирске Научно-исследовательского института информатики и вычислительной техники АПН СССР;

организация всесоюзного журнала «Информатика и образование»;

организация секции информатики в Ученом методическом совете Минпроса СССР;

написание II части пробного учебного пособия по информатике (X класс) и его массовое издание [17];

написание и издание методического пособия ко II части пробного учебного пособия [18];

публикация еженедельного «подвала» в «Учительской газете» под названием «ЭВМ всем, всем, всем» с методическими заметками по урокам информатики;

съемка 20 телевизионных уроков по курсу информатики и их трансляция по Центральному телевидению с общим объемом порядка 100 часов эфирного времени;

подготовка программы конкурсного стабильного учебника по информатике [19] и организация всесоюзного конкурса на учебник;

выработка санитарно-гигиенических требований к школьной вычислительной технике;

конструирование и предъявление межведомственным комиссиям моделей массовых школьных ПЭВМ «Корвет» и УКНЦ;

организация временного научно-техниче-

ского коллектива (ВНТК «Школа») для объединения усилий АН СССР, АПН СССР, министерств просвещения и высшего и среднего специального образования в проведении программы опережающих исследований и педагогических экспериментов по школьной информатике;

перевод и издание пробного учебного пособия и методических пособий к нему на языках советских республик.

Этот перечень дел и событий говорит сам за себя.

Настоящее

Содержание школьной информатики. Следует подчеркнуть, что школьная информатика — это конгломерат многих видов деятельности, группирующихся в *оконечную* деятельность, протекающую в основном в школе, и в *обеспечивающую* деятельность, происходящую главным образом за пределами школы.

Оконечная деятельность, в свою очередь, состоит из всех форм изучения информатики и из разнообразных применений ЭВМ как средства во всех сферах школьной жизни. Имеет смысл различать следующие виды применения ЭВМ в учебном процессе.

1. *Учебное* — использование ЭВМ в интересах преподавания конкретного школьного предмета с помощью специальных машинных программ, ориентированных на данный предмет.

2. *Орудийное* — компьютерная поддержка некоторых универсальных видов деятельности, органически входящих в учебный процесс, но не специфичных для того или иного предмета (письмо, рисование, черчение, ведение и использование информационных фондов, вычисления, коммуникация).

3. *Трудовое* — применение ЭВМ, связанное с профессиональной подготовкой и ориентацией. Оно охватывает как углубленное изучение информатики, так и пользовательское применение ЭВМ в разного рода профессиях или трудовых процессах внешкольного характера.

4. *Досуговое* — весь спектр неформального использования ЭВМ, диктуемого личными интересами (игры, увлечения, домашнее хозяйство и т. п.).

5. *Учительское* — применение ЭВМ в интересах учителя, начиная с организационной и контрольной поддержки в процессе урока и кончая разными видами домашней работы учителя.

6. *Организационное* — информационно-вычислительная поддержка управления школой и образованием.

7. *Дефектологическое* — все виды компьютерной поддержки обучения детей, обладающих дефектами и недостатками развития. В значительной мере вбирает в себя прочие виды школьного применения ЭВМ, но подчиняет их специальным требованиям дефектологии.

8. *Педвузовское*. Имеются в виду все аспекты применения ЭВМ в педвузе. Эта деятельность не касается непосредственно школьников, но моделирует все аспекты информатики и рассматривается в тесной взаимосвязи с ней.

Обеспечение школьной информатики характеризуют следующие ключевые понятия.

I. Наука
II. Методика
III. Учебники
IV. Компьютеры
V. Программы
VI. Школа (инфраструктура школьной информатики, формируемая в школе)
VII. Система (инфраструктура внешкольного управления делами школьной информатики)
VIII. Подготовка (молодых специалистов по школьной информатике)
IX. Переподготовка (рекрутирование специалистов по школьной информатике из других областей)
X. Ориентация (работников сферы просвещения, не являющихся специалистами по информатике)

Таким образом, реальное содержание школьной информатики — матрица, измерениями которой являются перечисленные ключевые виды оконечной деятельности (8 позиций) и обеспечивающей деятельности (10 позиций).

Рабочее определение информатики. Разработка научных основ курса информатики потребовала выработки некоторого соглашения о его предмете. Надо сказать, что за последние несколько лет, особенно в связи с организацией в АН СССР Отделения информатики, вычислительной техники и автоматизации, проходила довольно оживленная дискуссия по этому вопросу. Ниже дается некоторое рассуждение о предмете информатики, в основе которого лежит статья, подготовленная для энциклопедического словаря, и которое, как кажется автору, является равнодействующей большей части суждений, высказанных в дискуссии.

Информатика — это находящаяся в процессе становления наука, изучающая законы и методы *накопления, передачи и обработки информации с помощью ЭВМ*, а также (в переносном смысле) область человеческой деятельности, связанной с применением ЭВМ. Автор уравнивает понятия «информатика» и «вычислительная наука», хотя некоторые исследователи пытаются найти методологические различия между европейской «информатикой» и американо-английской «вычислительной наукой».

Предмет информатики существовал и развивался задолго до появления ЭВМ, однако либо покрывался другими науками, либо оставался в стороне от научного анализа. Появление ЭВМ, стремительное их развитие и применение актуализировало проблематику информатики, выявило ее своеобразие, придало ее проблемам масштаб, сформировало большой контингент новых специалистов. Современная информатика сложилась в недрах математики и кибернетики, системотехники и электроники, логики и лингвистики.

Основные научные направления информатики образуют такие дисциплины, как теоретические основы *вычислительной техники*, статистическая *теория информации*, теория *вычислительного эксперимента, алгоритмика, программирование, искусственный интеллект и информология*. Здесь под алгоритмикой понимается математический раздел информатики, посвященный анализу, формальным преобразованиям, построению и логическому выводу алгоритмов. При таком подходе в программировании концентрируются более инженерные вопросы реализации алгоритма при заданных пространственно-временны́х ограничениях средствами конкретного языка программирования с учетом всего жизненного цикла программного продукта. Информология занимается проблемами передачи и обработки информации в отдельных социумах и человеческом обществе в целом.

Информация, являясь общенаучной категорией, сближает некоторые разделы информатики с философией, а в остальные разделы информатики входит посредством более специфических понятий информационной и математической моделей, данных, алгоритма и программы. Используемое при этом понятие ЭВМ, компьютера, процессора относится не только к тем или иным техническим машинам, но и к любому физически реализованному или мысленному процессу формальной обработки информации. Это позволяет применять идеи и методы информатики для изучения информационного взаимодействия в биологических, социальных и вообще любых природных системах.

Прикладная информатика, или информатика как деятельность, обслуживает науку, технику, производство и другие виды человеческой деятельности путем создания и передачи в общество *информационной технологии*. Необходимость эффективного и повсеместного освоения этой новой технологии ставит перед всеми видами образования масштабные задачи распространения компьютерной грамотности и содействия ее перерастанию в информационную культуру общества.

Представление об информационной культуре. В уже упомянутом «манифесте» школьной информатики [4] формула информационной культуры была показана через призму так называемого «программистского стиля» мышления. Сейчас встает непростая задача выработки представления об информационной культуре, которое позволило бы избежать крайностей примитивного ремесленничества и снобистского профессионализма в обучении информатике. При всем при том это понятие должно быть достаточно емким, широким и конкретным, чтобы, не поступаясь своим фундаментальным содержанием, оно в то же время подводило учащихся к выбору адекватного уровня овладения информатикой в послешкольный период. Определенный подход к формуле информационной культуры был дан в пояснительной записке к конкурсной программе курса информатики [19]:

«Курс «Основы информатики и вычислительной техники» должен формировать у учащихся:

навыки грамотной постановки задач, возникающих в практической деятельности, для их решения с помощью ЭВМ;

навыки формализованного описания поставленных задач, элементарные знания о методах математического моделирования и умение строить простые математические модели поставленных задач;

знания основных алгоритмических структур и умение применять эти знания для построения алгоритмов решения задач по их математическим моделям;

понимание устройства и функционирования ЭВМ и элементарные навыки составления программ для ЭВМ по построенному алгоритму на одном из языков программирования высокого уровня;

навыки квалифицированного использования основных типов информационных систем (и пакетов прикладных программ общего назначения — *А. Е.*) для решения с их помощью практических задач и понимания основных принципов, лежащих в основе функционирования этих систем;

умение грамотно интерпретировать результаты решения практических задач с помощью ЭВМ и применять эти результаты в практической деятельности.

Эти требования, взятые в их минимальном объеме, составляют задачу достижения первого уровня компьютерной грамотности, а в максимальном объеме — перспектив-

ную задачу — воспитание информационной культуры учащихся».

Этот подход многим может показаться перегруженным, недостаточно фундаментальным и чересчур программистским. Тем не менее к его достоинствам следует отнести практичность, емкость и честность в признании факта, что хороший программист — пока наиболее надежный носитель информационной культуры, не выраженной еще в общечеловеческих категориях.

Школьная информатика на местах. Количество ЭВМ в школе стремительно нарастает, их производство и поставки находятся под жестким государственным и партийным контролем. Из доклада секретаря ЦК КПСС М. В. Зимянина в июле 1986 г. [21]:

«Главное сейчас, что требуется школам,— это помощь в подготовке кадров специалистов и в укреплении материально-технической базы. Большая ответственность здесь ложится на министерства электронной и радиотехнической промышленности, которые должны поставить в двенадцатой пятилетке школам и профтехучилищам около 500 тыс. компьютеров».

ЭВМ в школе используется сейчас в основном для поддержки курса информатики. Преимущество отдается Бейсику, Фокалу, прежде всего из-за их доступности на имеющихся машинах; интенсивно используется Е-практикум. На «Агатах» все чаще используется система «Школьница» с учебно-производственным языком «Рапира».

Применение ЭВМ в преподавании других предметов еще довольно долго будет носить фрагментарный и пробный характер. Количество сертифицированных учебных программ исчисляется единицами, но должно достичь к 1990 г. нескольких десятков тем, что в масштабах страны будет означать их доступность для каждого учащегося в среднем несколько раз в год по некоторому предмету.

Орудийное применение ЭВМ реально будет носить в ближайшее время демонстрационный характер. Универсальные пакеты прикладных программ начнут распространяться в 1988 г. Как обычно, на первом месте идут системы обработки текстов, за ними — машинная графика. О широком применении баз данных и электронных таблиц говорить еще рано.

Трудовое применение ЭВМ концентрируется, главным образом, в межшкольных кабинетах и там, где уже существуют прочные связи с базовыми организациями. Таких школ насчитывается несколько сотен.

Досуговое применение ЭВМ носит пока фрагментарный и некоммерческий характер, однако уже налицо элементы подпольного бизнеса. Здесь можно ожидать полезную самодеятельность молодежных организаций.

О сколько-нибудь широком учительском, организационном и дефектологическом применениях ЭВМ в школе говорить пока не приходится. Педвузовское применение ЭВМ сдерживается недостатком вычислительной техники и разработанных практикумов.

Обеспечение школьной информатики. Научная работа развертывается сейчас в ВНТК «Школа», в научных подразделениях АПН и ряде других вузов. Продолжается уточнение общей стратегии школьной информатики. Обсуждается вопрос о возможном смещении курса информатики в неполную среднюю школу и выработка в связи с этим научных основ профессиональной подготовки школьников в X—XI классах. Ведется работа по обоснованию спецификации нового поколения школьных ЭВМ. Существенным моментом научного анализа является различение двух поколений учебного применения ЭВМ: (1) инфильтративное внедрение ЭВМ в виде отдельных тематических программ в существующий курс по предмету и (2) «интегрированный» курс с органическим и непрерывным использованием ЭВМ. В настоящее время преобладает научная разработка обучающих программ 1-го поколения (их классификация и определение наиболее приоритетных тем). Весьма актуальным является определение номенклатуры орудийного применения ЭВМ в школе и формирование научной программы дефектологического применения ЭВМ. Становление школьной информатики настоятельно требует идентификации ее разделов в смежных науках: психологии, математике, языкознании и социологии.

Методика. Уже сейчас по стране насчитывается несколько десятков активных групп (педвузы, ИУУ и некоторые вузы общего профиля), в которых разрабатывается методическое обеспечение курса информатики. В ближайшие годы следует ожидать стабилизации методики по курсу информатики в IX—X классах, создания методических основ трудового применения ЭВМ и методов интеграции орудийного применения ЭВМ в преподавании разных предметов.

Учебники. Пробное учебное пособие по информатике будет работать еще год-два. Сейчас заканчивается подготовка второго, переработанного, издания объединенного учебника информатики IX—X, которое придет в школу в 1988/89 учебном году. Тем временем будет проходить доработка, апробация и внедрение новых учебников информатики на период 1990—1995 гг., в частности, на основе рукописей, хорошо зарекомендовавших себя на недавно состоявшемся конкурсе.

Основные учебники дополняются рядом вспомогательных изданий. Готовится энциклопедический словарь по информатике. Объявлен педвузовский учебник, ожидаются монографии по учебному и орудийному применениям ЭВМ. В издательстве «Детская литература» вышла превосходная энциклопедия Р. Свореня по современной радиоэлектронике [22].

Программное обеспечение. В качестве минимального штатного обеспечения школьных компьютеров на ближайшее время определены Е-практикум, системы программирования на Бейсике и Рапире, системы обработки текстов, машинной графики, ведения баз данных и электронных таблиц для орудийного применения ЭВМ. Создается система сертификации обучающих программ. В НИИ информатики и вычислительной техники АПН СССР формируется фонд педагогических алгоритмов и программ.

Переподготовка и ориентация. В НИИ информатики и вычислительной техники АПН СССР подготовлены программа и лабораторный практикум для 36-часового курса ориентации административных работников сферы просвещения, а в МГУ — для 36-часового курса информатики для учителей; они рассчитаны на преподавание в кабинетах, оборудованных ПЭВМ «Ямаха».

Будущее

Для страны такого масштаба, как СССР, значительную часть развития будет составлять экстенсивное распространение персонала, компьютеров, программного обеспечения и других ресурсоемких компонентов школьной информатики. Эта сторона работы является объектом государственной политики и планирования и определяется многими факторами. Если, однако, экстраполировать на 90-е годы темп, взятый в 12-й пятилетке, то к 2000 г. советская школа должна достигнуть первого уровня насыщения вычислительной техникой (один компьютер на 10—15 учащихся). Эта временная оценка задает тот диапазон, с которым следует соотнести анализ структурных изменений школьной информатики, неизбежных в серии переходов от возможного к желаемому.

Информатизация общества. Компьютеризация школы — это всего лишь часть общеисторического процесса, который сейчас получает название информатизации [23]. Под ней понимается одно из проявлений современного периода развития человеческого общества, состоящего в объективном усилении роли достоверного, исчерпывающего и опережающего знания во всех видах человеческой деятельности. В своей основе информатизация предопределяется исчерпанием возможностей экстенсивного развития производительных сил, осознанием ограниченности естественных ресурсов среды человеческого обитания, ростом сложности производственных отношений, появлением глобальных проблем, стоящих перед человечеством в целом и требующих осознания его единства на основе нового социально-политического мышления.

В широком смысле информатизация означает повышение степени информированности общества в целом и каждого его члена в отдельности, формирование общественного сознания на основе научно обоснованной и достоверной картины мира. При этом важна не просто пассивная доступность любой информации, а правильное и своевременное ее использование в гармоничном сочетании интересов общества и личности.

Технической основой информатизации являются компьютеризация и всемерное развитие систем связи. Их основные формы:

встраивание ЭВМ и микропроцессоров в разного рода существующие и создаваемые машины;

создание автоматизированных рабочих мест на базе ПЭВМ;

конструирование роботов;

построение автоматических систем управления работой машин и производств;

создание автоматизированных систем организационного управления и принятия решений;

безбумажное делопроизводство;

создание автоматизированных информационных систем и вычислительных центров коллективного пользования;

построение автоматических систем наблюдения за природными явлениями естественного и антропогенного происхождения;

создание глобальных систем связи и распространения массовой информации.

Существенным компонентом компьютеризации и информатизации является дополнительная форма отчуждения человеческого знания, его активизация и использование как непосредственной производительной силы в виде программного обеспечения и машинных банков данных и знаний.

Очевидно, что школьная информатика должна развиваться в неразрывной связи с историческим процессом информатизации.

Школа настоящего и школа будущего. Осознание этой связи выдвигает одну важную научную проблему. В течение сравнительно длительного времени (15—20 лет) школьная информатика будет развиваться в недрах сложившейся системы народного образования. Имеются в виду ее органи-

зационная структура и текущая методико-педагогическая доктрина: предметно-урочная система обучения, классно-кабинетное группирование учащихся, существующее разграничение познавательной и трудовой деятельности, упор на обучение так называемым «основам наук», концентрация учебного процесса в стенах школы и т. п.

Все эти институты образования слишком устойчивы, чтобы на них посягать; надо приспосабливаться к ним. Примерно десятилетнее опережение школьной информатики на Западе демонстрирует устойчивость указанных или аналогичных институтов образования, несмотря на существенно более широкий спектр стилей обучения. В то же время имеется немало убедительных технико-социологических прогнозов о несравненно более глубоком потенциальном влиянии новых информационных технологий на способы получения образования. Поэтому преобладание осторожных форм перестройки учебного процесса под воздействием ЭВМ должно сопровождаться опережающими и альтернативными исследованиями и экспериментами, которые позволят не упустить ни одну идею, какой бы необычной она вначале ни казалась. Это означает, что программа исследований, в частности, для ВНТК «Школа», должна обязательно предусматривать определенные футурологические разработки в поисках новой парадигмы образования, которая берет полную информатизацию общества в качестве предпосылки, а не в качестве грядущей конечной цели.

Пример продуктивности такого рода «опережающих» идей я мог бы найти в нашем собственном опыте. Когда в середине 70-х гг. мы взяли в качестве отправной точки проекта по школьной информатике гипотезу неограниченного доступа учащегося к информационно-вычислительным мощностям, это заранее обрекало любое методическое решение на неприемлемость в тогдашних условиях остаточного выделения ресурсов на образование. Когда же колокол пробил, коллективу специалистов было чем ответить.

Перспективные проблемы. В качестве одного из кандидатов на опережающие разработки естественно назвать вопросы компьютерного образования и воспитания в начальной школе (6—10 лет). С одной стороны, очевидна необходимость и интересность исследований и осторожных экспериментов по этой проблеме, с другой стороны, еще преждевременно формулировать какие бы то ни было гипотезы для массовой деятельности, относящейся к этой возрастной группе. Постепенное вовлечение шестилеток в общее образование, несомненно, будет способствовать поддержанию нужного тонуса в этой области исследований. Хотелось бы ожидать выхода этих исследований в область массовой педагогической практики к 14-й пятилетке. Здесь большие ожидания возлагаются на вклад со стороны специалистов по психологии развития.

Более близкой проблемой является смена поколений обучающих программ, состоящая в переходе к «интегрированным курсам», ведущим предмет в неразрывной связи с предметным, орудийным и трудовым применением ЭВМ и, стало быть, рабочей эксплуатацией основных идей и умений информатики.

Наиболее естественным кандидатом для отработки такого подхода является интегрированный курс информатики и математики IV—VIII классов 11-летней школы.

В группе школьной информатики СО АН СССР уже накоплен начальный опыт систематического знакомства с основами информатики с IV класса. В то же время до сколько-нибудь выраженной интеграции еще далеко, и проведение широких исследований в этом направлении весьма актуально. К концу 13-й пятилетки естественно ожидать разработки научных основ подобного рода курса.

В качестве весьма срочной и актуальной задачи на ближайшее будущее следует назвать развертывание широким фронтом программы исследований по дефектологическим применениям ЭВМ.

Школьная информатика на рубеже тысячелетий. Из разнообразия прогнозов, мнений и оценок, относящихся к 2000 г., постараемся выделить наиболее актуальные и правдоподобные.

Следует ожидать повсеместного преподавания курса информатики с долей времени непосредственной работы на ЭВМ в 50—60 %. Стабильный учебник по интегрированному курсу информатики для IV—VIII классов должен быть апробирован и готов к массовому применению.

Школа должна располагать несколькими тысячами сертифицированных обучающих программ по отдельным темам и разделам с покрытием всех предметов. Каждый учащийся должен иметь возможность регулярных выходов на ЭВМ для применения этих программ по каждому предмету.

Орудийное применение ЭВМ должно быть доступно каждому учащемуся, составляя до 30 % общего времени работы с ЭВМ, выделяемого на учащегося.

Школа должна выпускать не менее полутора миллионов человек с адекватной подготовкой по информатике, вычислительной технике и смежным с ними профессиям.

Досуговое применение ЭВМ должно быть доступно каждому в разных формах: домашние ЭВМ, вечерние кружки в школе, компьютерные клубы, досуговые центры.

Большая часть учителей получит начальный опыт личного применения ЭВМ, все учителя информатики и еще примерно столько же учителей-предметников будут регулярно применять ЭВМ в своей работе.

Следует ожидать, что к 2000 г. школа перейдет на компьютерный документооборот, осуществив интеграцию автоматизированных систем управления «Школа» и «Образование».

Все дефектологические школы и классы должны быть адекватно оборудованы техническими средствами и полным комплектом обучающих программ.

Педагогические вузы выйдут на общегосударственный уровень обеспечения вычислительной техникой, инструментальным и педагогическим программным обеспечением.

Научные исследование должны привести к появлению экспериментальных школ с новыми формами учебного процесса. Будет завершена апробация интегрированных курсов по основным предметам. К этому времени сложится модель полной информатизации общества и будут найдены общеобразовательные основы новых трудовых процессов.

Методическое обеспечение интегрированных курсов и применения ЭВМ в начальной школе станет на прочную основу. Сложатся методические основы альтернативных форм учебного процесса.

В школе должно быть размещено 4—5 млн. школьных ПЭВМ 2-го поколения вместе с их базовым программным обеспечением.

Естественно, что к этому времени инфраструктура школьной информатики получит свое полное развитие. В частности, для каждой школы будут подготовлены или переподготовлены по два-три учителя информатики, по два-три заведующих кабинетами вычислительной техники и не менее 200 тыс. учителей-предметников специально для ведения интегрированных курсов.

Литература

1. *Ершов А. П.* Системы программирования для мини- и микро-ЭВМ.— Всемирный электротехнический конгресс, 21—25 июня 1977. М., 1977. Секция 7, докл. 46, С. 1—10.

2. *Ершов А. П.* Персональная ЭВМ — предок млекопитающих в динозавровом мире ВЦКП.— В кн.: Материалы Всесоюзной научно-технической конференции по проблемам создания индивидуальных диалоговых систем на базе микро-ЭВМ «Диалог-82-микро»: персональные компьютеры (23—25 ноября 1982 г., Пущино).— Пущино, 1983, С. 9—25.

3. *Ершов А. П.* Автоматизация работы служащих // Микропроцессорные средства и системы. 1984. № 2. С. 6—15, 27.

4. *Ершов А. П., Звенигородский Г. А., Первин Ю. А.* Школьная информатика (концепции, состояние, перспективы). Новосибирск, ВЦ СО АН СССР, препринт № 152, 1979.

5. *Ершов А. П.* Программирование — вторая грамотность / ЭКО. 1982. № 2. С. 143—156.

6. *Ершов А. П., Звенигородский Г. А., Первин Ю. А., Юнерман Н. А.* ЭВМ в школе: опыт формулирования национальной программы. В кн.: INFO 84, 6—10. Februar 1984, Dresden, Plenarvorträge, 1. S. 53—63.

7. Основные направления реформы общеобразовательной и профессиональной школы // Правда. 1984. 14 апреля. С. 3—4.

8. В Центральном Комитете КПСС и Совете Министров СССР // Комсомольская правда. 1984. 29 апреля. С. 1—2.

9. В Политбюро ЦК КПСС // Известия. 1985. 30 марта. С. 1.

10. Программа для средних учебных заведений. М., Просвещение. 1985.

11. *Ершов А. П., Монахов В. М., Бешенков С. А. и др.* Основы информатики и вычислительной техники. Часть I. М., Просвещение, 1985.

12. *Ершов А. П., Монахов В. М., Кузнецов А. А. и др.* Изучение основ информатики и вычислительной техники. Часть I. М., Просвещение, 1985.

13. *Варсанофьев Д. В., Кушниренко А. Г., Лебедев Г. В.* Е-практикум — программное обеспечение школьного курса информатики и вычислительной техники // Микропроцессорные средства и системы. 1985. № 3. С. 27—32.

14. *Иоффе А. Ф.* Массовые персональные ЭВМ серии «Агат» // Микропроцессорные средства и системы. 1984. № 1. С. 56—60.

15. *Денисенко А.* Испытания КУВТ-86 // Информатика и образование. 1986. № 2. С. 69—73.

16. *Звенигородский Г. А., Глаголева Н. Г., Земцов П. А. и др.* Программная система «Школьница» и ее реализация на персональных ЭВМ // Микропроцессорные средства и системы. 1981. № 1. С. 50—55.

17. *Ершов А. П., Монахов В. М., Кузнецов А. А. и др.* Основы информатики и вычислительной техники. Часть II. М., Просвещение, 1986.

18. *Ершов А. П., Монахов В. М., Витиньш М. В. и др.* Изучение основ информатики и вычислительной техники. Часть II. М., Просвещение, 1986.

19. Программа курса «Основы информатики и вычислительной техники» // Микропроцессорные средства и системы. 1986. № 2. С. 86—89.

20. *Ершов А. П.* Компьютерный всеобуч // Учительская газета. 1984. 11 сентября. С. 2.

21. Школьной реформе — глубину и динамику действий. Совещание в ЦК КПСС // Учительская газета. 1986. 3 июля. С. 1—2.

22. *Сворень Р.* Электроника шаг за шагом. М., Детская литература. 1986.

23. *Ракитов А.* Информатизация общества и стратегия ускорения // Правда. 1987. 23 января. С. 2—3.

В. БЕЛОШАПКА
Канд. физ.-мат. наук

О языках, моделях и информатике

Третий год в наших школах ведется преподавание основ информатики. Прояснились основные проблемы этого курса — дефицит подходящей для школы вычислительной техники, сложности с созданием учебного программного обеспечения, подготовкой учительских кадров; стали видны пути их разрешения. Уже не за горами время, когда актуальная задача — достижение компьютерной грамотности — будет в принципе решена. Компьютер войдет в учебный процесс, работа по совершенствованию преподавания всех предметов будет вестись с учетом его возможностей. В недалеком будущем основные компоненты компьютерной грамотности (навыки работы с вычислительной техникой; элементарные навыки постановки, алгоритмизации и решения задач с помощью ЭВМ) будут приобретаться на занятиях по традиционным школьным предметам в их компьютеризированных вариантах, а более глубокое и серьезное обучение программированию, видимо, останется прерогативой специализированных курсов трудового обучения как элемента предпрофессиональной подготовки. Итак, не сегодня и, быть может, не завтра, но все же встанет вопрос: каковы задачи информатики как школьного предмета в этих новых условиях?

Ответ на него зависит от ответа на другой вопрос: что такое информатика? Что составляет предмет изучения этой науки, в чем особенность ее метода, каково ее место в системе научных дисциплин? Ни среди научной общественности, ни среди ее наиболее авторитетных представителей единого мнения по этим вопросам нет. Информатика находится в стадии становления. Среди высказываемых о ней мнений отметим следующие:

1. Информатика — это computer science, т. е. как реально существующей науки ее нет, а существует околокомпьютерная деятельность и порожденный ею конгломерат инженерных дисциплин, решающий задачи разработки ЭВМ и программного обеспечения, а также их применения, не имеющий собственного теоретического ядра, а заимствующий свои концепции из других наук, прежде всего из математики.

2. Информатика — это теоретическое программирование.

3. Информатика — это ряд разделов математики, а именно вычислительная математика, дискретная математика и другие, получившие в последнее время большое прикладное значение.

4. Информатика — это новое название кибернетики, т. е. науки об управлении.

5. Информатика — наука, изучающая методы фиксации, организации, хранения и использования научных знаний (то, что связывается с термином cognitiv science).

6. Информатика — наука, изучающая процессы сбора, хранения и обработки информации.

Мы далеки от того, чтобы утверждать, что все это неверно. К обсуждению этих точек зрения мы вернемся, но после того, как сформулируем еще одну, которая представляется нам наиболее фундаментальной.

Прежде чем изложить ее, сделаем несколько замечаний общего характера. Сознавая всю важность процесса накопления знаний и научной деятельности, следует отметить, что оно играет подчиненную роль по отношению к задаче создания целостной картины мира. В свете этого ситуация, сложившаяся в науке к началу XX в., может квалифицироваться как неблагополучная: на фоне нарастающего потока новых открытий и теорий целостная картина мира распалась. Почему это произошло? Кратко можно ответить так. Парадигма научного исследования, возникшая в эпоху Ренессанса и окончательно оформившаяся в Новое время, например, в работах Декарта и Бэкона, делавшая упор на объективизацию познания, породила нарастающую лавину открытий, но она же привела к нарушению равновесия между частными исследованиями и работой интегрирующего и синтетического плана. Хотя в это время в философско-методологическом плане велась большая работа и выдвигались весьма солидные концепции, например философские системы Гегеля и Канта, увенчавшиеся революционным переворотом в научно-философской картине мира — созданием Марксом и Энгельсом диалектического материализма, однако на уровне конкретных исследований баланс центробежности и центростремительности все же был нарушен.

Итак, необходимость в интегрирующем факторе общенаучного значения была очень острой, в частности, и потому, что тогда же проявилось новое в европейской науке обстоятельство: развитие частных наук, каждая из которых шла своим путем, привело к нарастанию непонимания между представителями разных наук и даже представителями различных школ внутри одной дисциплины. Было утрачено языковое единство.

На этом фоне и возникла некая центростремительная сила. В 1916 г. вышла работа швейцарского ученого Ф. де Соссюра «Основы общей лингвистики». В то время лингвистика представляла собой совокупность отдельных дисциплин, каждая из которых изучала тот или иной конкретный язык. Соссюр же исходил из того, что задачей лингвистики является изучение общеязыковых закономерностей. В своей работе он рассматривал язык как некоторую формальную систему, делая принципиальное разграничение между понятиями «речь» и «язык» и ограничивая именно формальной языковой системой предмет новой дисциплины. Для целей своего исследования Соссюр разработал новый понятийный аппарат. На богатейшем конкретно-языковом материале он убедительно доказал эффективность и плодотворность своей концепции, демонстрируя открытые им общеязыковые инварианты. Таким образом, взгляд на язык как на формальную систему и сопряженный с этим взглядом понятийный аппарат стал достоянием лингвистики, породив в ней особое направление — структурную лингвистику.

Но этим дело не ограничилось. Постепенно рядом исследователей начал осознаваться тот факт, что подход, предложенный Соссюром для описания языка, не менее плодотворен и эффективен при изучении самых разных объектов: биологических, этнических, экономических, технических и т. д. Не претендуя на полноту и не касаясь оценки полученных результатов, приведем список авторов, в тех или иных областях применявших и развивавших этот метод. Это Л. С. Выготский [1], В. Я. Пропп [2], О. М. Фрейденберг [3], Л. В. Канторович [4], Н. Винер [5], Дж. фон Нейман [6], Л. фон Берталанфи [7], А. А. Ляпунов [8], А. Н. Колмогоров [9], К. Шеннон [10], Н. А. Бернштейн [11], П. К. Анохин [12], Н. Хомский [13], К. Леви-Стросс [14] и другие — список можно продолжить. Могущество метода казалось безграничным. Почти любая проблема, любой объект, будучи описанными на новом понятийном языке, порождали формализованные модели, для исследования которых стали привлекать математические методы. Создавалось впечатление, что, наконец, найдены те элементарные структуры, из которых конструируется живое, мышление, общество. Да и само живое лишалось своего особого статуса, потому что наиболее увлекающиеся адепты новой науки брались создать гомункулуса в пробирке. Не было лишь подходящей пробирки, но вскоре появилась и она — заработали первые ЭВМ.

ЭВМ имели для новой науки принципиальное значение. Появилась возможность воплощать «в железе» формально-языковые модели, накопленные к тому моменту. Возникло новое направление исследований — работы по созданию «искусственного интеллекта», представители которого брались за решение таких задач, как машинный перевод, распознавание образов и другие, причем в ближайшие годы. Когда очередные сроки истекали, неудачи объяснялись техническими трудностями: нехваткой быстродействия и объемов запоминающих устройств. С тех пор прошло больше тридцати лет, быстродействие и объем памяти ЭВМ возросли на много порядков, но эти задачи в первоначальной, принципиальной постановке остались нерешенными. Однако попытки были не бесплодны. В результате уточнения и корректировки исходных задач кристаллизовалось представление о том, что всякий изучаемый нами объект (например, некая профессиональная деятельность) допускает вычленение своего формализуемого компонента, который можно адекватно описать в системно-информационных терминах и, следовательно, моделировать на ЭВМ. За это время сформировалась определенная культура постановки и решения задач с помощью ЭВМ, что привело даже к появлению новой профессии: системный аналитик, или когнитолог.

Итак, системно-информационный подход лишился статуса «теории всего», на который он претендовал, однако порожденный им язык получил общенаучное распространение, прежде всего в науках «мягкого» типа. На нем пишутся статьи по биологии, геологии, экологии, психологии и т. д., причем процесс его распространения шел порой независимо от основного русла развития информатики. Например, исследования Вернадского в геологии или Введенского и Кеннона в биологии носят выраженный системно-информационный характер. Более того, хотя освоение системного языка произошло в этих науках лишь отчасти, почти все наиболее впечатляющие открытия XX в. формулируются именно в его терминах. К примеру, в биологии отметим работы Уотсона и Крика по структуре ДНК, Селье по стрессу, Бэрнета по клеточной иммунологии.

Что же касается «точных» наук, то с ними системно-информационный взгляд на вещи связан еще основательнее. Открытие Соссюра произошло не на пустом месте; как на формирование дихотомии «речь — язык» оказали влияние работы Канта, так в основных понятиях системного метаязыка мы ви-

дим прямую преемственность с центральными понятиями основанной в XIX в. в работах Галуа и Абеля ветви математики, которая занимается изучением операций и структур и называется современной алгеброй. Например, понятие «инвариант» получает глубокое и правильное осмысление именно в групповом контексте. Отметим также значение теории множеств (Кантор) и успехов математической логики (Гильберт, Гедель) для становления формального взгляда на язык, а также для понимания принципиальных ограничений чисто формальной точки зрения. Что же касается физики, то, как хорошо известно, аппарат теоретической физики XX в.— это аппарат теории групп, как дискретных, так и непрерывных.

Таким образом, эта грандиозная трансформация языка науки явилась событием общенаучного масштаба, в ее формировании так или иначе участвовали основные концепции всех научных дисциплин — от философии (смысл — знак), математики (алгебраическая структура, формальный язык) до биологии и геологии (гомеостаз, биосфера). Краткий исторический очерк, приведенный нами, следует рассматривать как попытку нарисовать лишь общий контур этого процесса.

Вернемся к информатике. Науку принято характеризовать предметом изучения и методом; теперь мы готовы определить их. Предметом изучения информатики является вышеупомянутый новый системно-информационный язык, предназначенный для описания моделей (и, соответственно, сами эти модели как конструкции языка), а методом (определяющим ее применения) — формализация. Причем, как всякая наука, информатика имеет свое теоретическое ядро и свой прикладной компонент, который акцентирует инструментальное значение этого языка, т. е. способы его использования для описания тех или иных объектов. Что касается метода данной науки, то под формализацией понимается выделение формального компонента изучаемого объекта. Отметим, что «детские болезни», которые имели место в информатике 50—60-х гг., были связаны с допущением, что всякий объект сводится к своему формальному компоненту, или, в более мягкой формулировке, допускает адекватное формальное представление. Ситуация вполне аналогичная той, что сложилась в естествознании XVIII в. в связи с распространением механистической концепции.

Вернемся к изложенным в начале статьи точкам зрения на информатику, оценивая их с позиций предлагаемого нами подхода. Первая (computer science) отклоняется нами в негативной своей части (наука «информатика» все же есть!) и принимается как взгляд на прикладную информатику. Именно в этом, прикладном, точнее, аппаратном, аспекте следует оценивать роль прикладной информатики.

Теоретическое программирование находится по отношению к информатике в особом положении. Во-первых, оно довольно полно ассимилировало все основные ее концепции, во-вторых, занимаясь вопросами разработки программного обеспечения ЭВМ, оно оказывает определяющее воздействие на те разделы информатики, которые связаны с реализацией и исследованием формальных моделей на ЭВМ. Однако следует подчеркнуть, что, сводя концептуальный аппарат информатики к его реализации в сфере программирования, мы лишаем этот аппарат присущей ему универсальности.

Третья точка зрения справедлива в том смысле, что генетически информатика зависит от математики, от ее языка и формальных конструкций; однако если объекты, изучаемые математикой, формальны изначально, то сфера влияния информатики — моделирование объектов исходно неформальных средствами формализованных языков.

Взгляд на информатику как на кибернетику является ограниченным в силу существования большого класса систем, представляющих безусловный интерес для информатики и не являющихся управляемыми ни в каком разумном смысле (простейший пример — периодическая система Менделеева).

Отличие сформулированного нами подхода к информатике от пятой точки зрения (cognitiv science) в программистских терминах можно описать так: если последняя представляет предмет информатики как бы некоторой универсальной базой знаний, то мы считаем, что ее предмет — формализованный язык для описания и реализации таких, а также других систем.

Шестая точка зрения (процессы обработки информации) представляется нам не проясняющей суть дела, а, в силу ассоциаций с бытовым пониманием информации как всякого, в том числе и неформального, сообщения, затемняющей эту суть. Впрочем, можно рассматривать наш подход к информатике как уточнение и пояснение этой концепции.

Следует сделать еще одно уточнение. При поверхностном рассмотрении проблемы может создаться впечатление, что информатика является наукой философско-методологического плана. Оно обусловлено тем, что информатика имеет свое отношение ко всем основным конкретным исследованиям и свою точку зрения на них. Однако принципиальной особенностью ее метода, как было отмечено, является ориентация на исследование формального компонента изучаемого объекта, тогда как философско-методологические науки занимаются исследованием общих закономерностей содержательного плана. Однако в той мре, в какой в том или ином философском исследовании возникают и исследуются формальные модели, в нем возможно инструментальное использование языка и методов информатики. К примеру, исследование экономической системы капитализма, предпринятое Марксом, в той части, где он рассматривает модели экономических отношений, носит системно-информационный характер; однако здесь информатика играет подчиненную роль, подобную роли математики в физике. Отметим также, что указанное различие наиболее рельефно выступает при констатации различий между системным методом, как методологической установкой, и информатикой.

Вернемся к обсуждению педагогических аспектов проблемы.

Нужно ли преподавать информатику, понимаемую так, как было сформулировано выше, в средней школе?

Возможно ли это, а если да, то как?

На первый вопрос мы отвечаем положительно, и вот почему. Информатику как научную дисциплину породил языковой общенаучный кризис начала XX в., на преодоление которого она была ориентирована; таким же образом появление информатики в ряду школьных дисциплин могло бы решить задачу восстановления ослабевших и утраченных межпредметных связей, выработки у учащихся широкого, универсального взгляда на всю совокупность изучаемых наук, воссоздания у них единой научной картины мира. Кроме того, наш подход к информатике позволяет привнести в этот учебный предмет то теоретическое ядро, которое способно сцементировать прикладные аспекты курса. Рассмотрим, наконец, вопрос о компьютерной грамотности. Решение задачи с использованием ЭВМ имеет три аспекта, и, соответственно, есть три группы трудностей, которые при этом преодолеваются. Во-первых, необходима соответствующая техника; во-вторых, подходящее программное обеспечение; в-третьих, должны быть решены задачи вычленения в данной предметной области ее формализуемого компонента, его формального описания и алгоритмизации. Анализ современного состояния этих проблем показывает, что затраты на решение задачи с помощью ЭВМ распределяются так: техника — самая дешевая часть, программное обеспечение занимает промежуточное положение, а основные трудности и затраты приходятся на конкретную постановку задачи и формализацию. В соответствии с этими тремя аспектами работа ведется инженером по вычислительной технике, программистом и системным аналитиком. Кем же из них будет выпускник нашей школы? Вообще говоря — никем, он — тот четвертый, предметный специалист (врач, администратор, педагог, инженер и т. д.), в сотрудничестве с которым системный аналитик будет решать задачу перевода содержательных представлений предметной области на язык формализованных моделей и описаний. При этом врач или администратор может иметь весьма приблизительное представление об устройстве компьютера и принципах программирования, но владение основами системно-информационного подхода для успешного выполнения работы ему необходимо: именно это позволит системному аналитику, освоившему азы данной предметной области, вести с ним плодотворное сотрудничество. Вот почему именно на изучение основ языка системно-информационных представлений, причем на широком предметном материале, и должен быть направлен курс школьной информатики.

Ну а что можно сказать о возможности такого курса? Докажет ее, конечно, только экспериментальная проверка, однако априори ясно, что шансы на успех есть. Попробуем дать обзор материала, который можно было бы привлечь к составлению курса. Приведем его в форме программы, но подчеркнем, что он таковой не является.

1. Введение. Что такое информатика. Отношения «объект — модель». Место информатики в системе научных и инженерных дисциплин, ее предмет и метод.

Основные понятия: модель, система, элемент, отношение.

Практические занятия: использование графов для описания и элементарного анализа простейших систем; работа на ЭВМ с конструктором электрических схем.

2. Системы классификационного типа. Систематика в биологии, таблица Менделеева, организация и ведение библиотечного каталога. Общая характеристика задач классификации: анализ данных, систематизация знаний. Автоматическая классификация, распознавание образов.

Основные понятия: структура, иерархия, классификация, свойство, процедура.

Практические занятия: работа на ЭВМ со справочно-информационной системой, системой распознавания образов.

3. Управление и динамические системы. Примеры биологических, экологических, технических систем. Гомеостаз, адаптация, задачи автоматического регулирования. Управление производством, информационная технология.

Основные понятия: гомеостаз, адаптация, обратная связь, информация, биосфера, состояние системы, управление.

Практические занятия: работа по исследованию компьютерных моделей (биологической, экологической, технической, оптимизационной).

4. Конечные автоматы. Формальные нейронные сети, персептрон, ЭВМ. ЭВМ как универсальный конечный автомат (ЭВМ как универсальная моделирующая среда). История возникновения ЭВМ, поколения ЭВМ. Задачи автоматизации человеческой деятельности.

Основные понятия: дискретная система, конечный автомат, представление информации, вход — выход, запоминающее устройство, процессор, преобразование информации.

Практические занятия: работа с конструктором логических схем, реализованным на ЭВМ.

5. Языковые системы. Противопоставление «речь — язык». Язык как формально-знаковая система. Формальная грамматика, проблема контекста. Задача автоматического перевода.

Основные понятия: язык, знак, алфавит, синтаксис, грамматика, контекст, метафора.

Практические занятия: семантический анализ волшебной сказки, синтаксический анализ текста, работа с предметным тезаурусом, решение лингвистических задач.

6. Элементы программирования. Алгоритм, исполнитель, программа, ЭВМ как универсальный исполнитель. Язык программирования и его основные конструкции. Типы данных.

Основные понятия: алгоритм, исполнитель, команда, серия, ветвление, цикл, процедура, массив.

Практические занятия: работа по составлению простейших программ (20—30 команд) на учебном языке программирования.

7. Заключение. История становления основных идей и методов информатики. Системно-информационный язык описаний моделей, его основные конструкции. Метафоричность системно-информационного языка. Аналогия как научный метод. Ограничения формального подхода. Системно-информационный метод решения задач с помощью ЭВМ. Способы использования ЭВМ в народном хозяйстве, типы решаемых задач и используемого при этом программного обеспечения. Информатика и компьютеризация общества.

Основные понятия: СУБД, АСУ, САПР, ГАП, АОС, информационная технология.

Практические занятия: решение содержательно поставленной задачи на ЭВМ с реализацией основных этапов (корректная постановка, формализация и моделирование, алгоритмизация, программирование, анализ результата).

Основой методики преподавания данного курса должно быть систематическое использование системно-информационного языка для описания и исследования формальных моделей из разных предметных областей. В приведенном варианте курс ориентирован на старшие (IX—XI) классы, однако, не меняя методической установки, а только изменив список изучаемых моделей, можно переориентировать его на средние (VII—VIII) классы. При этом можно шире использовать те модели, которые изучались в общеобразовательных предметах. Преподавание такого курса в младших классах представляется нецелесообразным в связи с формальным характером его содержания, не соответствующим особенностям младшего школьного возраста.

В заключение следует отметить, что построенный таким образом курс слабо зависит от своего конкретного наполнения и поэтому может включать любой учебный материал; однако *превалирование* любой конкретной прикладной области (будь то системы хранения знаний или программирование) обедняет курс и в какой-то мере лишает его одного из основных достоинств: возможности продемонстрировать учащимся наличие в мире универсальных закономерностей.

Литература

1. *Выготский Л. С.* Избранные психологические исследования. М., 1956.
2. *Пропп В. Я.* Морфология сказки. М., 1928.
3. *Фрейденберг О. М.* Миф и литература древности. М., 1978.
4. *Канторович Л. В.* Математические методы организации и планирования производства. М., 1939.
5. *Винер Н.* Кибернетика. М.: Сов. радио. 1969.
6. *Фон Нейман Дж.* Теория самовоспроизводящихся автоматов. М.: Мир, 1966.
7. *Von Bertalanfy L.* The theory of open systems // Science, 1950. 111.
8. *Ляпунов А. А.* О некоторых общих вопросах кибернетики. М.: Физматгиз, 1957.
9. *Колмогоров А. Н.* Теория передачи информации // Труды сессии АН СССР по вопросам автоматизации производства. М., 1957.
10. *Шеннон К.* Работы по теории информации и кибернетики. М.: Мир, 1963.
11. *Бернштейн Н. А.* Очерки по физиологии движений и физиологии активности. М., 1966.
12. *Анохин П. К.* Очерки по физиологии функциональных систем. М., 1975.
13. *Хомский Н.* Синтаксические структуры. Новое в лингвистике. М.: ИЛ, 1962.
14. *Levi-Strauss C.* Structural Antropology. N. Y., 1967.

А. ШЕНЬ

Информатика в IX классе

Вспомогательные алгоритмы и их использование

Знакомство со вспомогательными алгоритмами проще всего начать с примеров, связанных с рисованием на плоскости. В предыдущих номерах журнала мы говорили об исполнителе Черепашке и составляли программу рисования слова МИР. В ней естественно выделялись части, соответствующие рисованию букв М, И, Р. Использование вспомогательных алгоритмов позволяет «узаконить» это разделение, явно отразив его в самом тексте алгоритма.

```
алг написание слова МИР
нач
    буква М
    шаг вправо
    буква И
    шаг вправо
    буква Р
кон
алг буква М
нач
    вперед (4)
    налево (30)
    назад (2)
    направо (60)
    вперед (2)
    налево (30)
    назад (4)
кон
алг буква И
нач
    вперед (4)
    назад (4)
    направо (30)
    вперед (4.8)*
    налево (30)
    назад (4)
кон
алг буква Р
нач
    вперед (4)
    направо (90)
    вперед (2)
    направо (90)
    вперед (20)
    направо (90)
    вперед (2)
кон
алг шаг вправо
нач
    не рисуй
    направо (90)
    вперед (1)
    налево (90)
    рисуй
кон
```

Смысл алгоритмов, использующих другие алгоритмы (называемые вспомогательными), может быть определен различными способами (однако в рассмотренном примере между всеми этими способами не проявляются различия; к трудностям, возникающим в более сложных примерах, мы еще вернемся).

Способ 1. Текстовая подстановка. Можно считать, что перед исполнением алгоритма, содержащего ссылки на вспомогательные алгоритмы (такие ссылки называют «вызовами» вспомогательных алгоритмов), эти ссылки «раскрываются». Это значит,

* *Продолжение*. Начало см. Информатика и образование. 1987. №№ 3—5.

* Мы пишем «4.8» вместо «4,8», так как запятая используется в алгоритмическом языке для разделения аргументов вспомогательного алгоритма.

что строка с названием вспомогательного алгоритма заменяется его текстом. Если мы проделаем это в нашем примере, то получим тот самый алгоритм рисования слова МИР, который приведен в предыдущих номерах журнала.

Способ 2. Разные исполнители. Можно представить себе дело иначе. Вообразим, что у нас имеется Помощник, в обязанности которого входит исполнение вспомогательных алгоритмов (в данном примере — алгоритмов «буква М», «буква И», «буква Р», «шаг вправо»). Мы можем, например, скомандовать ему «буква М», после чего Помощник сам, без нашего участия, командует Черепашкой в соответствии с алгоритмом рисования буквы М. После этого Помощник сообщает нам, что команда выполнена и он готов к исполнению следующей команды. При этом Помощник знает лишь алгоритмы рисования букв «М», «И», «Р» и алгоритм «шаг вправо», ему безразлично, как они используются в основном алгоритме: ему командуют — он выполняет. Наоборот, мы, командуя Помощником, можем уже не интересоваться деталями рисования букв.

Способ 3. Расширение системы команд. Будем считать, что наша Черепашка может учиться. Это значит, что ее систему команд можно расширять, обучая ее новым командам. Обучение состоит в том, что мы сообщаем ей алгоритм выполнения соответствующей команды. После этого она добавляет эту команду в свою систему команд. Так, сообщив Черепашке алгоритмы «буква М», «буква И», «буква Р» и «шаг вправо», мы получаем нового исполнителя — Обученную Черепашку с системой команд: вперед (x), назад (x), налево (x), направо (x), рисуй, не рисуй, буква М, буква И, буква Р, шаг вправо. После этого можно исполнять основной алгоритм «написание слова МИР» обычным образом — все его команды входят в систему команд исполнителя Обученная Черепашка.

Все сказанное, конечно, лишь наглядные образы, иллюстрирующие исполнение вспомогательных алгоритмов. Для изложения в классе можно выбрать тот, который кажется наиболее наглядным.

Зачем нужны вспомогательные алгоритмы? Пример использования вспомогательных алгоритмов при рисовании слова МИР вызывает естественный вопрос: а зачем все это нужно? Почему нельзя сразу вписать алгоритмы рисования букв и «шаг вправо» внутрь основного алгоритма? Что мы выигрываем, выделяя их? Ответить на это можно следующее.

1. Прежде всего, выделение вспомогательных алгоритмов облегчает составление и проверку алгоритмов; при составлении основного алгоритма (в нашем примере — алгоритма рисования слова МИР) мы можем уже не заботиться о том, как работают вспомогательные алгоритмы, а нужно знать только, что они делают. Наоборот, при составлении вспомогательного алгоритма (в нашем примере рисование букв, «шаг вправо») мы не должны учитывать, где и зачем он используется. Нужно лишь следить за тем, чтобы он действительно делал то, для чего предназначен. Благодаря этому разделению основной и вспомогательный алгоритмы могут составлять даже разные люди — им только следует договориться о том, что должен делать вспомогательный алгоритм (но не о том, как он будет это делать).

2. Использование вспомогательных алгоритмов позволяет сократить решение задачи. В самом деле, при «раскрытии» вызовов вспомогательного алгоритма «шаг вправо» его текст придется писать дважды. Правда, в нашем случае эта экономия компенсируется «накладными расходами» на оформление (строки **алг, нач, кон**). Однако в других случаях использование вспомогательных алгоритмов может привести к существенному упрощению — это бывает, когда один и тот же вспомогательный алгоритм используется многократно (если бы, например, нужно было написать слово МИМ, то экономия была бы больше, так как в этом случае алгоритм «буква М» используется дважды).

Такой прием сокращения длины текста не нов. По существу он же применяется, например, в книгах по математике, где формулы и теоремы нумеруют или снабжают именами, а затем пишут «формула (7)» или «теорема Пифагора» вместо того, чтобы повторять формулу или формулировку теоремы. Такие записи аналогичны обращению к вспомогательным алгоритмам.

Заметим кстати, что уменьшение длины алгоритма обычно позволяет уменьшить объем памяти ЭВМ, занимаемый им.

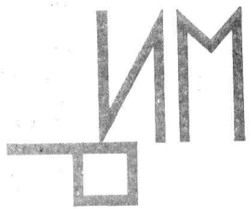
3. Использование вспомогательных алгоритмов полезно, если нам нужно решать несколько похожих задач. Например, если после слова МИР нам потребуется составить алгоритм написания слова РИМ, то все вспомогательные алгоритмы можно оставить без изменений, изменив основной алгоритм:

```
алг написание слова РИМ
нач
    буква Р
    шаг вправо
    буква И
    шаг вправо
    буква М
кон
```


Прежде чем читать дальше, остановитесь и подумайте: правилен ли этот алгоритм?

Оказывается, нет. Результат его работы изображен на рис. 1. Вспомним: после рисования буквы Р Черепашка остается в середине буквы, головой влево. После этого буквы И и М она будет рисовать повернутыми. Как же исправить ошибку?

1



Сформулируем более точно: как составить алгоритмы рисования букв так, чтобы из них можно было составлять алгоритмы рисования любых слов, чередуя рисование букв с командой «шаг вправо»?

Ответ таков: надо договориться о том, где находится Черепашка в начале и конце рисования каждой буквы. Будем считать, что каждая буква вписана в прямоугольник (рис. 2), начальное положение Черепашки — головой вверх в левом нижнем углу, а конечное — головой вверх в правом нижнем углу. Алгоритмы «буква М» и «буква И» удовлетворяют этому требованию (если быть точным, то в алгоритме «буква И» следует заменить 4.8 на $8\sqrt{3}$ — проверьте!), а вот алгоритм «буква Р» требует изменений. Сделаем их.

```
алг буква Р
нач
    вперед (4)
    направо (90)
    вперед (2)
    направо (90)
    вперед (2)
    направо (90)
    вперед (2)
    назад (2)
    не рисуй
    направо (90)
    назад (2)
кон
```

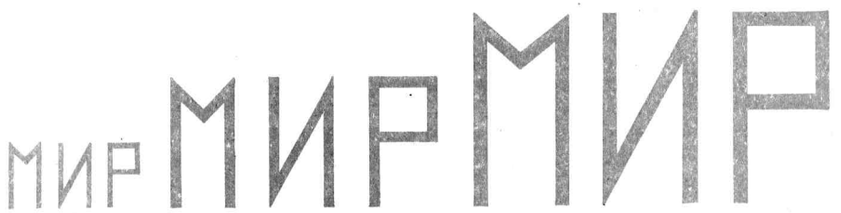
2

Заметим, что в наших требованиях к алгоритмам рисования букв ничего не сказано о том, рисует или не рисует Черепашка после их исполнения. В частности, после исполнения алгоритма «буква Р» Черепашка не рисует, а после исполнения остальных алгоритмов — рисует. Это не страшно (мы считаем, что повторное выполнение команды «не рисуй» допустимо).

Ошибка, допущенная при составлении алгоритма рисования слова РИМ, довольно типична. Она произошла из-за того, что при разделении задачи (рисование слова) на подзадачи (рисование букв) не были достаточно точно определены эти подзадачи: мы договорились о том, какие буквы должны рисовать вспомогательные алгоритмы, но не о том, где должна быть Черепашка до и после исполнения. Программисты сказали бы по этому поводу так: «спецификации алгоритмов рисования букв были недостаточно точными».

Вспомогательные алгоритмы с аргументами. Использование аргументов во вспомогательных алгоритмах можно пояснить на том же примере рисования слова МИР. Представим себе, что мы хотим это слово несколько раз подряд нарисовать с разным размером букв (рис. 3). Нужно ли составлять для этого три вспомогательных алгоритма рисования каждой буквы (для маленького, среднего и большого размеров)? Оказывается, нет. Вместо этого можно составить алгоритмы рисования букв в заданном масштабе d (число d показывает, во сколько раз буква крупнее, чем в приведенном выше алгоритме; так, $d=1/3$ означает, что все размеры буквы уменьшены в три раза). Заголовок алгоритма рисования буквы М будет выглядеть так:

3



алг буква М (вещ *d*)
арг *d*

(аналогично для других букв). Составим теперь основной алгоритм.

```
алг трижды МИР
нач
    буква М (0.25)
    шаг вправо
    буква И (0.25)
    шаг вправо
    буква Р (0.25)
    шаг вправо
    буква М (0.5)
    шаг вправо
    буква И (0.5)
    шаг вправо
    буква Р (0.5)
    шаг вправо
    буква М (1)
    шаг вправо
    буква И (1)
    шаг вправо
    буква Р (1)
кон
```

Выпишем теперь использованный здесь вспомогательный алгоритм рисования буквы М.

```
алг буква М (вещ d)
арг d
нач
    вперед (4*d)
    налево (30)
    назад (2*d)
    направо (60)
    вперед (2*d)
    налево (30)
    назад (4*d)
кон
```

(звездочка — знак умножения). Аналогично выглядят и алгоритмы рисования букв И и Р.

Упражнение. В построенном алгоритме интервалы между буквами не зависят от их размера. Как сделать так, чтобы между маленькими буквами были маленькие интервалы, а между большими — большие?

Проиллюстрируем теперь на этом примере некоторые способы исполнения вспомогательных алгоритмов. Если мы хотим перед исполнением основного алгоритма «раскрыть» вызовы вспомогательных алгоримов, то не следует переписывать их текст буквально (если это сделать, то в основном алгоритме будет фигурировать неизвестно чему равная величина *d*). Вместо этого при переписывании следует заменить букву *d* на число, стоящее в данном обращении к вспомогательному алгоритму. Так, строку «буква М (0.25)» надо заменить на

```
вперед (4*0.25)
налево (30)
назад (2*0.25)
направо (60)
вперед (2*0.25)
налево (30)
назад (4*0.25)
```

и т. д.

Другой способ — использование Помощника, исполняющего вспомогательные алгоритмы, — также несколько видоизменяется. В этом случае мы не просто командуем ему «буква М», но и сообщаем аргумент (к примеру, 0.25). После этого Помощник присваивает величине *d* значение 0.25 и приступает к исполнению алгоритма «буква М». Видя строку «вперед (4*d)», он умножает 4 на 0.25, получает 1 и командует Черепашке «вперед (1)», и т. д.

Вспомогательные алгоритмы с аргументами и результатами. Рассмотрим пример из УП: нахождение большего из трех чисел. Вот основной алгоритм:

```
алг БИТ (вещ a, b, c, y)
арг a, b, c
рез y
нач вещ z
    БИД (a, b, z)
    БИД (z, c, y)
кон
```

Вспомогательный алгоритм:

```
алг БИД (вещ a, b, c)
арг a, b
рез c
нач
    если a≥b
      то c:=a
      иначе c:=b
    все
кон
```

Рассмотрим разные варианты их исполнения.

Если мы «раскроем» обращения к вспомогательному алгоритму внутри основного буквально, то получим такой текст:

```
нач вещ z
    если a≥b
      то c:=a
      иначе c:=b
    все
    если a≥b
      то c:=a
      иначе c:=b
    все
кон
```

Ясно, что он не имеет никакого смысла (тут, в частности, вообще не фигурирует *z* — результат основного алгоритма).

Как же «раскрыть» вызовы правильно? Нужно, как говорят, «подставить фактические параметры вместо формальных». Это значит следующее. В описании вспомогательного алгоритма имеются величины *a*, *b*, *c*. Они называются формальными параметрами. В обращении к нему фигурируют другие имена величин — в первом обращении написано БИД (*a*, *b*, *z*), а во втором БИД (*z*, *c*, *y*). Величины, входящие в обращение к вспомогательному алгоритму, называются фактическими параметрами. Для разных вызовов они могут быть разными. Перед «раскрытием» вызова вспомогательного алгоритма следует заменить в тексте вспомогательного алгоритма формальные параметры на соответствующие им фактические. В первом обращении соответствие между формальными и фактическими параметрами таково:

формальные параметры *a*, *b*, *c*;
фактические параметры *a*, *b*, *z*.

Таким образом, при раскрытии имена *a* и *b* остаются, а *c* следует заменить на *z*:

```
если a≥b
  то z:=a
  иначе z:=b
все
```

Во втором обращении соответствие таково:

формальные параметры *a*, *b*, *c*;
фактические параметры *z*, *c*, *y*;

замена такова: *a*→*z*, *b*→*c*, *c*→*y*. Получаем:

```
если z≥c
  то y:=z
  иначе y:=c
все
```

Таким образом, после раскрытия вызовов вспомогательных алгоритмов получаем следующее:

```
алг БИТ (вещ a, b, c, y)
арг a, b, c
рез y
нач вещ z
    если a≥b
      то z:=a
      иначе z:=b
    все
    если z≥c
      то y:=z
      иначе y:=c
    все
кон
```

Проследив за работой этого алгоритма, мы убеждаемся, что он действительно помещает в *y* наибольшее из *a*, *b*, *c*: после исполнения первой команды ветвления *z* равно большему из чисел *a* и *b*, а после второй *y* равно большему из чисел *z* и *c*, т. е. большему из чисел *a*, *b*, *c*:

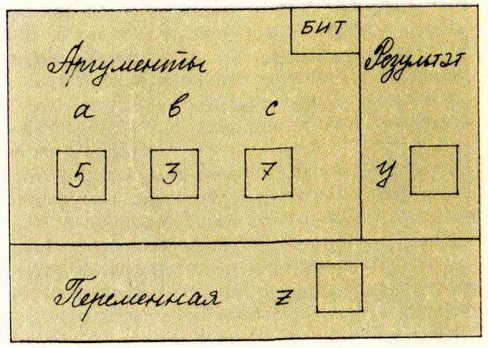
$\max(a, b, c) = \max(\max(a, b), c) = \max(z, c)$

Прежде чем перейти к другому способу исполнения, сделаем еще одно замечание. В данном случае во вспомогательном алгоритме не было промежуточных величин. Если же они есть, то случается, что их имена совпадают с именами величин, имеющихся в основном алгоритме. Чтобы из-за этого не возникла путаница, при подстановке текста вспомогательного алгоритма на место его вызова помимо замены формальных параметров на фактические необходимо предварительно изменить имена промежуточных величин вспомогательного алгоритма. Приведем соответствующий фрагмент из описания алгоритмического языка Алгол-60:

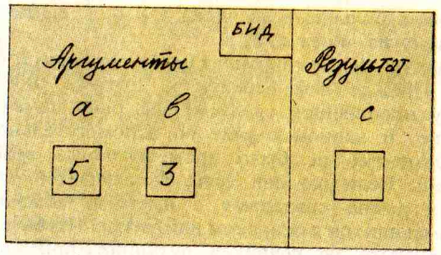
«Любой формальный параметр... повсюду в теле процедуры (вспомогательного алгоритма — *А. Ш.*) заменяется на соответствующий фактический параметр... Возможность противоречий между идентификаторами (именами величин — *А.Ш.*), вставляемыми в тело процедуры... и идентификаторами, уже присутствующими в теле процедуры... устраняется систематическими изменениями... идентификаторов, затронутых такими противоречиями» (мы пропустили в этой цитате части, которые непонятны без детального знакомства с Алголом-60).

Теперь опишем исполнение алгоритма БИТ, если алгоритм БИД исполняется отдельно. Представим себе учителя Тройкина, исполняющего алгоритм БИТ, и ученика Двойкина, умеющего исполнять алгоритм БИД.

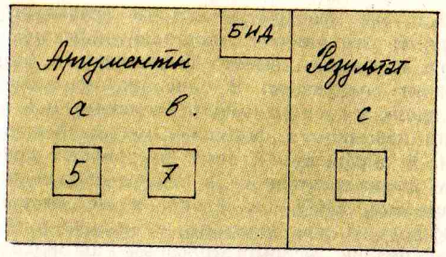
Начиная работу, учитель Тройкин рисует на доске картинку (рис. 4). Будем рассматривать исполнение алгоритма для аргументов 5, 3, 7. Эти числа записаны в рамках для аргументов. Рамки результатов и промежуточной величины пусты. Тройкин приступает к работе. Первой командой, которую нужно исполнить, является вызов



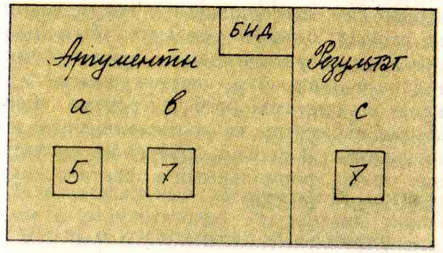
4



5



7



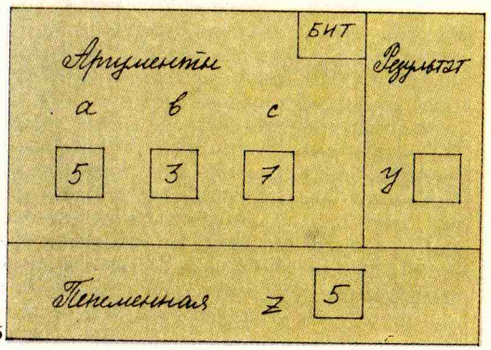
8

вспомогательного алгоритма БИД. И Тройкин командует: «Двойкин, к доске! Исполните алгоритм БИД для аргументов 5 и 3 и сообщите результат!» Двойкин выходит к доске (другой!) и принимается за работу (рис. 5). Обратите внимание, что рамки с надписями «*a*», «*b*», «*c*» на доске БИД не имеют ничего общего с рамками «*a*», «*b*», «*c*» на доске БИТ, кроме названия. В рамках «*a*», «*b*» доски БИД записаны числа, сообщенные Тройкиным Двойкину — числа 5 и 3 (они совпадают с записями в рамках «*a*» и «*b*» доски БИТ, но это случайность). Двойкин приступает к исполнению алгоритма БИД. Условие в команде ветвления ($a \geqslant b$) соблюдается, выполняется команда $c:=a$, записывающая в рамку *c* число 5. Двойкин говорит учителю: «Результат равен 5», стирает с доски и садится на место. Учитель записывает сообщенный Двойкиным результат в рамку *z*, на этом выполнение первой команды алгоритма БИТ заканчивается (рис. 6). Но Двойкину не удается спокойно посидеть на месте: следующая команда алгоритма БИТ — снова вызов вспомогательного алгоритма БИД. И Тройкин командует: «Двойкин, к доске! Исполните алгоритм БИД для аргументов 5 и 7 и сообщите результат!» Вызов БИД (*z*, *c*, *y*) происходит при $z=5$, $c=7$. Вызванный снова Двойкин рисует картинку (рис. 7), исполняет алгоритм (рис. 8), полученный результат сообщает Тройкину, стирает с доски и садится на место. Тройкин записывает его в рамку «*y*», и на этом исполнение алгоритма БИТ кончается (рис. 9).

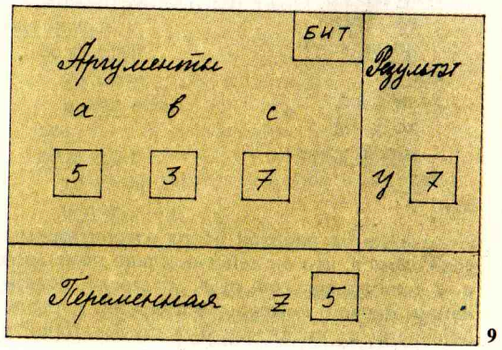
Оба разобранных варианта исполнения вспомогательных алгоритмов иллюстрируют такое свойство: имена величин, использованные во вспомогательном алгоритме, не имеют значения и могут быть выбраны произвольно; если бы мы написали

алг БИД (**вещ** α, β, γ)
арг α, β
рез γ
нач
  если α⩾β
    то γ:=α
    иначе γ:=β
  все
кон

(как это сделано в УП), в исполнении основного алгоритма ровным счетом ничего бы не изменилось. В частности, при выборе названий для величин вспомогательного алгоритма можно, как мы видели, использовать имена, используемые и в основном алгоритме. При этом одинаково именованные величины основного и вспомогательного алгоритмов — совершенно разные переменные.



6



9

Трудные случаи использования вспомогательных алгоритмов. Приведем два примера, в которых возможно различное толкование смысла вспомогательных алгоритмов.

Пример 1. Пусть основной алгоритм содержит фрагмент

.....
i:=2
F(*a*[*i*],*i*)
.....

(*i* — целая переменная, *a* — целочисленная таблица), а вспомогательный алгоритм F таков:

алг F (**цел** *p*, *q*)
рез *p*, *q*
нач
  q:=3
  p:=7
кон

Если раскрыть вызов вспомогательного алгоритма так, как мы это делали, то получится

i:=2
i:=3
a[*i*]:=7

т. е. *i* станет равным 3, элементу *a*[3] будет присвоено значение 7.

С другой стороны, если мы рассмотрим вариант с двумя исполнителями, получится вот что. Исполнитель основного алгоритма делает *i* равным 2, затем командует «исполнить алгоритм F и сообщить результаты». Результаты будут равны 7 и 3. Исполнитель основного алгоритма выполняет присваивания *a*[*i*]:=7 (т. е. *a*[2]:=7, так как в этот момент *i*=2) и *i*:=3. Итак, *i* станет равным 3, а *a*[2] станет равным 7. Получается, что такой способ исполнения приводит к другим результатам.

Пример 2. Пусть основной алгоритм содержит строку F(*a*, *a*) (*a* — целая переменная), а вспомогательный алгоритм таков же, как и в предыдущем примере. Тогда при раскрытии вызова получим

a:=3
a:=7

и окончательное значение *a* будет равно 7.

Если же мы попытаемся проследить за исполнением алгоритма двумя исполнителями, то обнаружим, что исполнитель основного алгоритма говорит: «Исполнить алгоритм F и сообщить результаты», ему сообщают «7, 3», и теперь он стоит перед неразрешимой задачей: нужно в рамку «*a*» записать 7 и 3 одновременно.

А теперь изменим вспомогательный алгоритм, переставив в нем строки:

алг F1 (**цел** *p*, *q*)
рез *p*, *q*
нач
  p:=7
  q:=3
кон

Казалось бы, алгоритмы F и F1 с точки зрения результатов их работы неотличимы. Однако при раскрытии вызовов F (*a*, *a*) и F1 (*a*, *a*) мы получим:

для F	для F1
a:=3	*a*:=7
a:=7	*a*:=3

и это уже совсем не одно и то же.

После рассмотрения этих примеров у читателя может возникнуть вопрос: как же правильно исполнять вспомогательные алгоритмы? Какие результаты должны получаться «на самом деле»?

Однако вопрос этот сформулирован неверно. Смысл тех или иных конструкций алгоритмического языка определяется его разработчиками, так что правильная постановка вопроса такова: какой способ исполнения вспомогательных алгоритмов имели в виду разработчики? УП оставляет его открытым.

Все эти трудности, скорее всего, останутся не замеченными школьниками, тем не менее полезно иметь их в виду (хотя, конечно, не стоит касаться их на уроках, если школьники сами не обратят на них внимания).

Упорядочение линейной таблицы

Задача упорядочения линейной таблицы состоит в следующем: имеется целочисленная таблица *a*[1:1000]; надо ее упорядочить, т. е. переставить числа в таблице так, чтобы выполнялось $a[1] \leqslant a[2] \leqslant \ldots \leqslant a[1000]$ (отличие от формулировки задачи в УП состоит в том, что размер таблицы фиксирован и равен 1000; о причинах такого рода изменений мы говорили в предыдущих номерах журнала). Заметим, что в условии задачи нельзя заменить «⩽» на «<», так как в таблице *a* могут встречаться одинаковые элементы.

Для начала изложим решение, следующее схеме из УП (с изменениями, вызванными тем, что все таблицы имеют фиксированную длину).

Какой элемент должен стоять после упорядочения первым? Ясно, что минимальный. А вторым? Тоже ясно — минимальный из оставшихся (кроме первого). Это наводит на такую идею алгоритма: будем заполнять новую (упорядоченную) таблицу от начала к концу. Введем переменную *k*, равную числу уже поставленных на свои места эле-

ментов. Точнее, мы будем требовать соблюдения такого условия:

первые k элементов таблицы a уже стоят на своих местах (т. е. совпадают с первыми k элементами упорядоченной таблицы, которую мы хотим построить) (1)

Вначале $k=0$; постепенно k будет увеличиваться. Когда k станет равным 1000, упорядочение закончено; все элементы стоят на своих местах. Впрочем, упорядочение можно закончить при $k=999$: если первые 999 элементов стоят на своих местах, то и последний тоже — для него осталось единственное место.

Итак, схема алгоритма такова:

```
k:=0
пока k≠1000
нц
  увеличить k на 1, не нарушая условия (1)
кц
```


Осталось понять, как увеличить k на 1, не нарушая (1). Пусть первые k мест таблицы уже заполнены правильно, и нам надо, чтобы были заполнены правильно первые $k+1$ место. Другими словами, нужно поставить на $k+1$-е место нужный элемент. Какой? Наименьший среди элементов таблицы, не считая первых k. Запишем это.

```
k:=0
пока k≠1000
нц
  положить s равным номеру минимального
  элемента среди a[k+1], ..., a[1000]
  поменять местами k+1-й и s-й элементы
  k:=k+1
кц
```

Выделим два указанных действия во вспомогательные алгоритмы.

```
алг минэлемент (цел таб a[1:1000], k, s)
арг a, k
рез s
```

Он исполняется при $k<1000$; после его исполнения s равно номеру минимального элемента среди $a[k+1], ..., a[1000]$.

```
алг обмен (цел таб a[1:1000], i, j)
арг a, i, j
рез a
```

Он исполняется при $1\leqslant i, j\leqslant 1000$; после его исполнения значения элементов $a[i]$ и $a[j]$ меняются местами; при $i=j$ таблица a не меняется.

Теперь можно составить основной алгоритм.

```
алг упорядочение (цел таб a[1:1000])
арг a
рез a
нач цел k, s
  k:=0
  пока k≠1000
  нц
    минэлемент (a, k, s)
    обмен (a, k+1, s)
  кц
кон
```

Теперь составим вспомогательные алгоритмы. Первый из них аналогичен приведенному в предыдущем номере журнала.

```
алг минэлемент (цел таб a[1:1000], k, s)
арг a, k
рез s
нач цел t
  t:=k+1
  s:=k+1
  пока t≠1000
  нц
    t:=t+1
    если a[t]<a[s]
      то s:=t
    все
  кц
кон
```

Идея второго алгоритма также нам уже знакома.

```
алг обмен (цел таб a[1:1000], i, j)
арг a, i, j
рез a
нач цел t
  t:=a[i]
  a[i]:=a[j]
  a[j]:=t
кон
```

Легко проверить, что при $i=j$ этот алгоритм работает правильно (не меняет таблицу a).

Задача решена. Отметим, что при ее решении мы использовали метод, который называют «методом последовательных уточнений», «пошаговой детализацией», «программированием сверху вниз» и т. д. Его суть в том, что мы сначала составляем основной алгоритм с использованием вспомогательных, не составляя самих вспомогательных алгоритмов, а лишь определяя, какие задачи они должны решать, и лишь потом переходим к составлению вспомогательных алгоритмов.

Заметим также, что в основном алгоритме (как мы уже говорили) можно заменить условие «$k\neq 1000$» на «$k\neq 999$».

Упорядочение линейной таблицы: другое решение. Это решение по своим характеристикам (длина алгоритма, скорость его работы и т. д.) весьма близко к уже изложенному. Тем не менее идея его поучительна. В качестве условия выберем такое:

первые k элементов таблицы стоят в возрастающем порядке: $a[1]\leqslant ... \leqslant a[k]$ (2)

Начальное значение k равно 1, конечное — 1000 (здесь уже 999 не годится!). Приходим к такой схеме алгоритма:

```
k:=1
пока k≠1000
нц
  увеличить k на 1, не нарушая условия (2)
кц
```

Как же увеличить k? Если $a[k]\leqslant a[k+1]$, то нам повезло и первые $k+1$ элементов случайно оказались в нужном порядке. Если же нет, то элемент $a[k+1]$ стоит правее, чем нам нужно и его надо сдвигать к началу таблицы, пока он не встанет на свое место. Другими словами, его надо менять местами с предыдущим элементом до тех пор, пока он не окажется больше или равен предыдущему элементу или пока он не попадет в начало таблицы. Обозначив через s «текущий» номер сдвигаемого элемента, получаем вот что:

```
алг упорядочение (цел таб a[1:1000])
арг a
рез a
нач цел k, s
  k:=1
  пока k≠1000
  нц
    s:=k+1
    пока s≠1 и a[s]<a[s—1]
    нц
      обмен (a, s—1, s)
      s:=s—1
    кц
    k:=k+1
  кц
кон
```

Алгоритм «обмен» уже приводился.

Здесь есть одна тонкость: что будет, если $s=1$ при проверке условия внутренней команды повторения? С одной стороны, условие $s\neq 1$ и $a[s]<a[s—1]$ вроде бы можно считать ложным, так как уже условие $s\neq 1$ ложно. С другой стороны, в условие входит часть $a[s]<a[s—1]$, которая при $s=1$ не имеет никакого смысла, так как $s—1=0$, а нулевого элемента в таблице нет. Поскольку способ проверки условий в УП не описан, можно придерживаться любой из приведенных точек зрения. Во втором случае можно видоизменить алгоритм, например, так:

```
нач цел k, s, лит готово
  k:=1
  пока k≠1000
  нц
    s:=k+1
    готово:=«нет»
  пока готово=«нет»
  нц
  если s=1
    то готово:=«да»
    иначе
      если a[s]<a[s—1]
        то
          обмен (a, s—1, s)
          s:=s—1
        иначе
          готово:=«да»
      все
    все
  кц
кон
```


Упорядочение таблицы: эффективный алгоритм. Приводимый ниже алгоритм упорядочения более сложен, чем разобранные, но и более эффективен. Можно убедиться, что используемый в нем метод позволяет упорядочить таблицу из n элементов за время, примерно пропорциональное $n\log n$, а время упорядочения этой же таблицы с помощью любого из предыдущих методов примерно пропорционально n^2 (мы говорим о «времени» работы алгоритма для краткости: следовало бы говорить о числе выполняемых действий — проверок условий и исполнения команд присваивания).

Идею эффективного алгоритма упорядочения (его называют «сортировка слиянием») можно пояснить наглядно следующим образом. Пусть у нас есть ящик с библиотечными карточками, которые нужно разложить по алфавиту. Сначала разложим все карточки в группы по одной карточке в каждой. Затем будем соединять группы попарно: из двух групп по одной карточке получится группа из двух карточек, которые мы расположим в алфавитном порядке. Снова соединим группы попарно, делая из двух групп по две карточки одну из четырех (в которой карточки также идут в алфавитном порядке). На следующих этапах мы получим группы по 8 карточек, затем по 16 и так далее до тех пор, пока все карточки не попадут в одну группу, внутри которой они будут идти в алфавитном порядке (если число карточек не является целой степенью двойки, то некоторые группы будут неполными).

При таком способе упорядочения важную роль играет процесс «слияния» двух упорядоченных стопок (групп) карточек в

одну. Его можно описать так:

пока хоть одна из исходных стопок непуста
нц
  выбор
    при обе стопки непусты:
      переложить ту из двух верхних карточек, которая идет по алфавиту раньше, в конец новой стопки
    при одна из стопок пуста:
      переложить верхнюю карточку другой стопки в конец новой стопки
  все
кц

Здесь использована команда выбора, описанная в УП для X класса; при желании ее можно заменить на команду ветвления.

Возвратимся теперь от карточек к задаче упорядочения таблицы. Схема алгоритма будет такова. Введем переменную k (длина упорядоченных участков) и будем требовать выполнения такого условия:

участки длины k, на которые разбивается таблица a (от начала к концу), упорядочены (3)

Например, при $k=2$ условие (3) означает: $a[1] \leqslant a[2]$, $a[3] \leqslant a[4]$, ...

Вначале $k=1$ (участки длины 1 упорядочены автоматически). В конце $k \geqslant 1000$, все элементы таблицы оказываются в одном участке. На каждом шаге слияния длина упорядоченных участков увеличивается вдвое. Итак:

$k:=1$
пока $k<1000$
нц
  увеличить k вдвое, не нарушая условия (3)
кц

Удвоение k без нарушения (3) требует слияния 1-го и 2-го участков, 3-го и 4-го и т. д. с сохранением порядка. Это слияние будет выполняться с помощью команды повторения. Введем переменную s, играющую роль границы между обработанной и необработанной частями: на отрезке $a[1]$, ... $a[s]$ упорядочены участки длиной $2k$, а на остальной части — длиной k. Вначале $s=0$. Схема удвоения k становится такой:

$s:=0$
пока $s<1000$
нц
  слить очередную пару участков, соответственно увеличив s
кц

Очередная пара участков — это $a[s+1]$... ...$a[s+k]$ и $a[s+k+1]$...$a[s+2k]$, если только мы не подошли уже к правой границе таблицы. Если же мы уже подошли к правой границе, то второй участок может быть короче или вообще отсутствовать. Запишем разные варианты действий в виде команды ветвления:

если $s+k+1 \leqslant 1000$ (второй участок непуст)
то
  слияние (a, s, $s+k$, min(1000, $s+2*k$))
  $s:=\min(1000, s+2*k)$
иначе (второй участок пуст)
  $s:=1000$
все

Здесь слияние (a, p, q, r) — вспомогательный алгоритм, работающий при $0 \leqslant p < q < r \leqslant 1000$ и соединяющий упорядоченные участки $a[p+1]...a[q]$ и $a[q+1]...a[r]$ в упорядоченный участок $a[p+1]...a[r]$.

Приведем теперь полный текст основного алгоритма.

алг сортировка слиянием (**цел таб** a[1:1000])
арг a
рез a
нач цел k, s
  $k:=1$
  пока $k<1000$
  нц
    $s:=0$
    пока $s<1000$
    нц
      если $s+k+1 \leqslant 1000$
      то
        слияние (a, s, $s+k$, min (1000, $s+2*k$))
        $s:=\min(1000, s+2*k)$
      иначе
        $s:=1000$
      все
    кц
    $k:=2*k$
  кц
кон

Осталось составить алгоритм слияния. Мы будем следовать описанной схеме с карточками. Введем переменные $i1$ и $i2$ — число карточек, взятых из первой и второй групп. «Взятые» карточки будем помещать не в саму таблицу a (чтобы они не перепутались со старыми), а во вспомогательную таблицу **цел таб** b[1:1000], начиная с $b[p+1]$. Вначале в первой группе $q-p$ карточек, во второй $r-q$ карточек. Первая карточка первой группы есть $a[p+i1+1]$, второй — $a[q+i2+1]$. Первая группа непуста, если $i1<q-p$, вторая — если $i2<r-q$. Место в новой группе для очередной карточки $b[p+i1+i2+1]$. В конце работы нужно переписать элементы $b[p+1]...b[r]$ в таблицу a; при этом используется переменная величина i — число переписанных элементов.

Заметим, что действия в четырех вариантах команды выбора бывают всего двух видов: перекладывание карточки из первой исходной группы в новую (варианты 1 и 3) и перекладывание карточки из второй исходной группы в новую (варианты 2 и 4).

алг слияние (**цел таб** a[1:1000], p, q, r)
арг a, p, q, r
рез a
нач цел $i1$, $i2$, i, **цел таб** b[1:1000]
  $i1:=0$; $i2:=0$
  пока $i1 \neq (q-p)$ **или** $i2 \neq (r-q)$
  нц
    выбор
      при $i1<q-p$ **и** $i2<r-q$ **и** $a[p+i1+1+1] \leqslant a[q+i2+1]$ (обе группы непусты, в первой карточка меньше):
        $b[p+i1+i2+1]:=a[p+i1+1]$
        $i1:=i1+1$
      при $i1<q-p$ **и** $i2<r-q$ **и** $a[p+i1+1+1] \geqslant a[q+i2+1]$ (обе группы непусты, во второй карточка меньше):
        $b[p+i1+i2+1]:=a[q+i2+1]$
        $i2:=i2+1$
      при $i1<q-p$ **и** $i2=r-q$ (первая группа непуста, вторая пуста):
        $b[p+i1+i2+1]:=a[p+i1+1]$
        $i1:=i1+1$
      при $i1=q-p$ **и** $i2<r-q$ (вторая группа непуста, первая пуста):
        $b[p+i1+i2+1]:=a[q+i2+1]$
        $i2:=i2+1$
    все
  кц
  $i:=0$
  пока $i \neq r-p$
  нц
    $a[p+i+1]:=b[p+i+1]$
    $i:=i+1$
  кц
кон

Завершая изложение эффективного алгоритма сортировки, отметим, что этот алгоритм довольно сложен — он заведомо сложнее всех алгоритмов, приведенных в УП (однако среди задач в конце УП есть и гораздо более сложные). Мы привели его для того, чтобы показать, что алгоритмы бывают весьма нетривиальными и что их составление — трудное, но зачастую увлекательное занятие.

Продолжение следует

Т. ПОДДУБНАЯ
Томский государственный университет

Запись алгоритмов на Бейсике

Первые шаги

Начнем с простой программы, которая читает два вещественных числа, присваивает их переменным A и B и меняет их местами, если A>B, а затем печатает результат своей работы. Для удобства справа от текста программы помещены английские слова, используемые в операторах, и их перевод на русский язык.

```
20 INPUT A, B              INPUT — ввод
25 IF A<=B THEN GO TO 45   IF — если
30 LET C=A                 THEN — то
35 LET A=B                 GO TO — иди к
40 LET B=C                 LET — пусть
45 PRINT A, B              PRINT — печатать
50 STOP                    STOP — останов
```

Что сразу обращает на себя внимание?

Прежде всего, нумерация операторов (команд) целыми числами.

Использование в записи команд ввода данных (INPUT) и вывода результатов (PRINT).

Хотя программа и понятна, она не похожа на изученный ранее способ оформления алгоритмов.

Совершенно очевидно, что операторы с номерами 30, 35 и 40 — операторы присваивания. Для их записи используется служебное слово LET и знак = (на некоторых ДВК слово LET можно опускать).

Оператор с номером 25 организует ветвление в зависимости от истинности условия A>B. Отметим, что он тоже отличается от команды ветвления типа **если—то—иначе.**

В программе используется так называемый оператор безусловного перехода (GO TO). Без него невозможно написать ни одну программу на Бейсике, в которой требуется организовать ветвление команд. Таков уж этот язык!

Вторая программа последовательно читает

```
10 пар вещественных чисел и заносит их
в переменные A и B. Для каждой пары она
меняет местами их значения в случае,
если A>B, и распечатывает результат.
10 LET K=1
15 IF K>10 THEN GO TO 60
20 INPUT A, B
25 IF A<=B THEN GO TO 45
30 LET C=A
35 LET A=B
40 LET B=C
45 PRINT A, B
50 LET K=K+1
55 GO TO 15
60 STOP
```

Попробуйте ответить на вопросы.

Каковы минимальное и максимальное значения переменной с именем K?

Сколько раз будет выполнен оператор безусловного перехода GO TO 60, входящий в состав команды с номером 15?

Сколько раз при работе программы будет выполнена команда с номером 55 и команда с номером 15?

После анализа второй программы ясно, что она представляет собой команду повторения; снова отметим, что ее оформление не совпадает с известной ранее формой **пока — нц — кц**.

Величины в программах на Бейсике

Назвался груздем —
полезай в кузов.

На гранях кубика приведена классификация величин по типу, возможному в языке Бейсик (вещественный, целый, литерный), и по характеру (постоянные, переменные). Уже отмечалось, что реализации Бейсика на различных ДВК отличаются друг от друга. Программист, прежде чем приступить к работе на конретном ДВК, должен познакомиться с возможностями языка Бейсик на нем. Так, если в теоретическом (американском) стандарте, описанном в книге Т.Уорта (*Т. Уорт.* Программирование на языке Бейсик. М.: Машиностроение, 1981), определены два типа данных — вещественный арифметический тип и строковый (литерный), то в языке Бейсик ДВК-2 арифметический тип подразделяется на целый и вещественный, а в Бейсике ДВК-1 отсутствует строковый тип и определен только один вещественный.

Описание величин, которое дается ниже, относится к языку Бейсик ДВК-2 по причине наибольшей полноты типов данных в этой реализации. Что же видно при первом взгляде на ЛОС 1?

На языке Бейсик возможно использование констант и переменных вещественного, целого и литерного типов, а также скалярных величин (простых переменных) и массивов. Поясним опорный сигнал «назвался груздем». Он говорит о том, что тип констант и переменных в программах на Бейсике задается видом константы или имени.

Собственно, константа во всех алгоритмических языках однозначно задается своим видом (значением), а вот переменные, как правило, требуют специального описания; в Бейсике же в таких описаниях нужды нет.

Длина имени в Бейсике ограничена. Оно не может содержать более двух символов. Имя — это либо одна буква, либо (в случае двух символов) буква и цифра. Это правило относится к именам вещественных переменных. Как видно из ЛОС 1 и приведенных ниже примеров, к имени целой переменной добавляется символ %. Этим же символом заканчиваются и целые константы.

К именам литерных переменных добавляется символ ¤. Литерные константы заключаются в кавычки.

Третье направление классификации, представленное на ЛОС — скалярные (простые) величины и массивы величин. Массив имеет не только тип, но и размер, поэтому он обязательно должен быть описан. Это делается в предложении следующего вида:

DIM имя массива (список верхних границ)

DIM — сокращенное от английского слова dimension (размер). Список верхних границ (он помещается в круглые, а не в квадратные скобки!) содержит максимальные значения индексов по каждому измерению описываемого массива. Они перечисляются через запятую. Нижние границы индексов в Бейсике всегда равны нулю и не указываются в описании (опорный сигнал — зачеркнутое слово НИГР).

В одном операторе DIM можно описать несколько имен массивов. Например,

DIM A(3,5), B(5,5), C(3,5), M %(9)

Здесь описаны четыре массива. Три из них вещественные (об этом говорят их имена), двухмерные, а один — целый одномерный. Таким образом, тип элементов массива определяется видом имени. Приведенные на ЛОС примеры поясняют это обстоятельство.

Обращение к элементу массива осуществляется с помощью переменной с индексами. Индексы помещаются в круглые скобки. Например, A¤(2) — третий по порядку элемент литерного массива, который мог быть описан в программе, например, как DIM A¤(16).

В языке Бейсик с арифметическими константами и переменными (как простыми, так и индексированными) допускаются все арифметические операции и операции сравнения: сложение +, вычитание —, умножение ×, деление /, возведение в степень^, операции сравнения =, <, >, <=, >=, < > (последний знак означает ≠). Для операции возведения в степень иногда используется другой знак (в зависимости от типа ДВК).

Почему вводятся специальные знаки для операций деления и возведения в степень?

Машина «не видит» надстрочных и подстрочных символов. Она читает все символы подряд, и потому программа на Бейсике, как и на любом другом языке программирования, должна быть записана в строку. Не должно быть выражений типа A^B или дробей $\frac{A}{B}$. Они с использованием введенных выше обозначений запишутся следующим образом: A^B и A/B.

Выражения вычисляются по известным правилам арифметики и алгебры в соответствии со старшинством операций:

возведение в степень — первый ранг;
умножение и деление — второй ранг;
сложение и вычитание — третий ранг;
сравнение — четвертый ранг.

Программист может пользоваться скобками (только круглыми) для организации необходимого порядка. В выражении без скобок подряд идущие операции одного ранга выполняются в порядке следования слева направо.

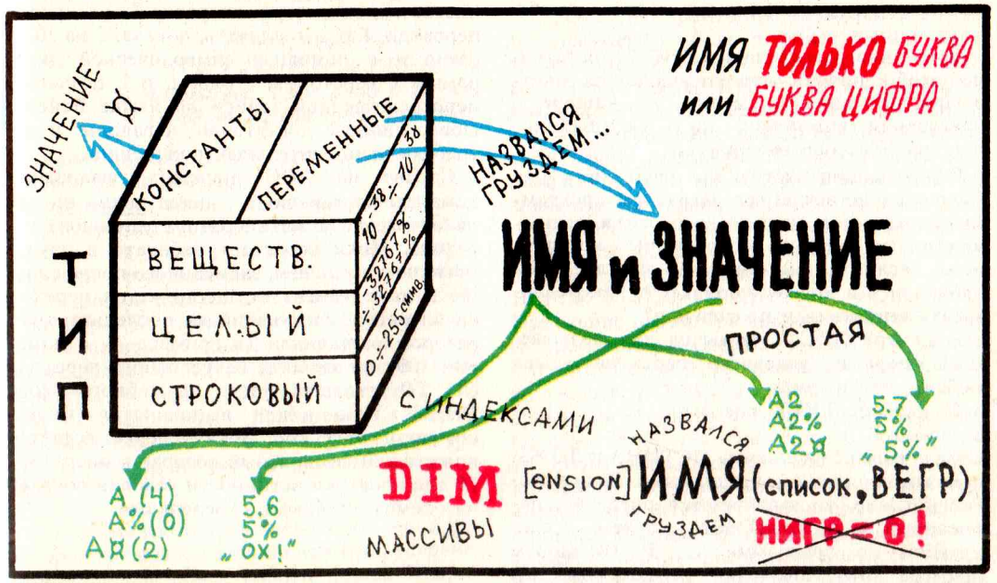
Со строковыми (литерными) величинами в языке Бейсик производятся операции сравнения и присоединения. Для обозначения операции присоединения используется символ + или &, например:

```
5 E¤=«СВЕТИТ»+«МЕСЯЦ»
6 PRINT E
7 E¤=«СВЕТИТ» &«ЯСНЫЙ»
8 PRINT E
```

После выполнения этого фрагмента программы на экране появятся две строки — СВЕТИТ МЕСЯЦ и СВЕТИТ ЯСНЫЙ.

Сравнение строковых данных друг с другом выполняется посимвольно слева направо в коде КОИ-7.

Операция, выполняемая над данными одного типа, дает результат того же типа.



	Значение константы	Имя переменной
Вещественной	3.44 344.E—02 —225 4. .75	A A3
Целой	20 % —345%	A% A3%
Литерной	«СВЕТИТ» «МЕСЯЦ»	A¤ A3¤

Управляющие инструкции на Бейсике

На безрыбье и рак — рыба.

Уже одного взгляда на ЛОС 2 достаточно, чтобы понять, что управляющих инструкций ветвления типа **если — то — иначе** и повторения типа **пока — нц — кц** в Бейсике нет (на ЛОС они зачеркнуты).

О реализации инструкции следования уже было сказано выше: все операторы программы нумеруются целыми числами и выполняются в порядке возрастания номеров, даже если на клавиатуре дисплея они набирались в другом порядке. А как реализовать ветвления и повторения?

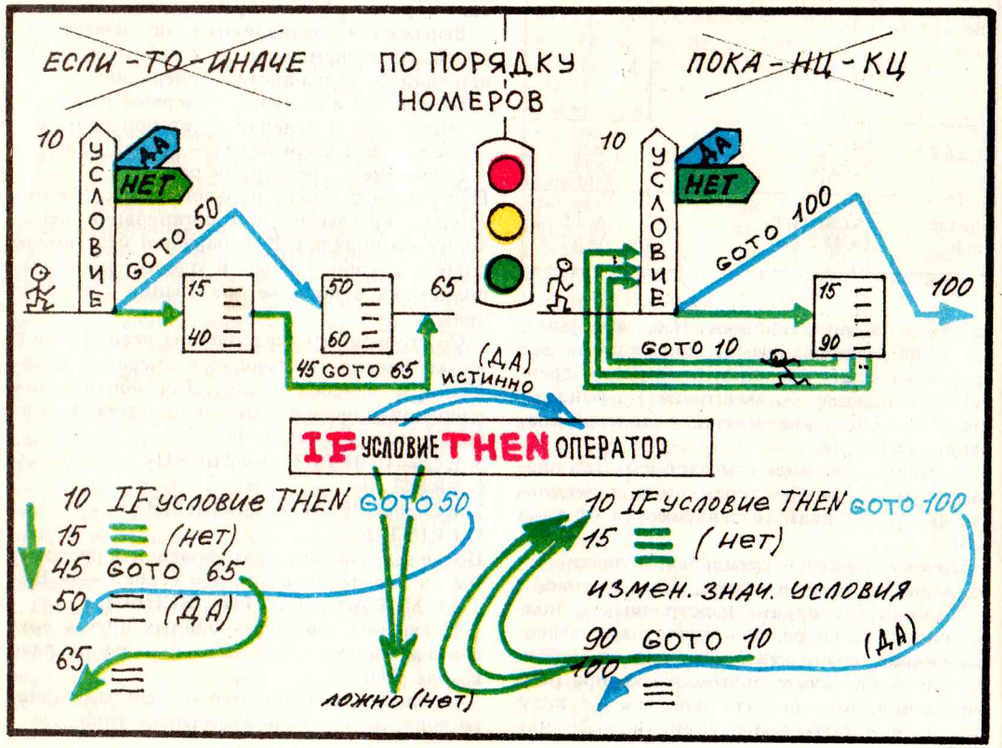
В центре ЛОС 2 помещен так называемый оператор условного перехода языка Бейсик. Его формат:

5 IF условие THEN оператор
6 . . . . . .

Стрелочками с надписями ИСТИНА и ЛОЖЬ показан порядок его выполнения. Если условие истинно, то выполняется оператор, написанный за словом THEN, а затем — следующий по программе за IF (в нашем примере это оператор с номером 6). Если же условие ложно, то сразу выполняется оператор, следующий за оператором IF. Таким образом, требующейся нам развилки на **то** и **иначе** нет, но ее можно организовать, использовав оператор безусловного перехода. Как это сделать, показано на ЛОС слева и с помощью символической диаграммы с верстовым столбом, и с помощью непосредственной записи на языке Бейсик. Оба варианта достаточно наглядны и не требуют дополнительных пояснений.

Справа на ЛОС показана реализация команды повторения — цикла **пока—нц—кц** на основе того же оператора условного перехода. Цикл перестает работать в случае истинности условия, записанного после слова IF. В этот момент осуществляется переход на оператор, следующий за последним оператором тела цикла (в примере — с номером 100) по команде безусловного перехода GO TO, стоящей после служебного слова THEN. Тело цикла выполняется многократно до тех пор, пока условие остается ложным. Многократный возврат к оператору IF для проверки истинности условия показан на схемах тройными стрелочками.

ЛОС 2



Цикл FOR (для) на языке Бейсик

Коси, коса, пока роса;
Роса долой — и ты домой.

...упрямы вы, и все одно и то же
твердить вам надобно сто раз!

Анализ циклических алгоритмов позволяет разбить их на два класса.

1. Известно точное число повторений какой-то группы команд. Например, необходимо написать алгоритм вычисления средней утренней температуры осеннего месяца. Пусть для определенности это будет октябрь, в котором 31 день. Нам необходимо сначала сложить 31 значение, каждое из которых является результатом измерения температуры в один из дней октября. Таким образом, число повторений команды присваивания, вычисляющей сумму температур, в этой программе заранее известно и равно 31.

2. Неизвестно точное число повторений некоторой группы команд, но известно условие, определяющее конец работы цикла. Например, для тех же результатов измерения температуры требуется определить, в какой день октября была зарегистрирована в первый раз температура ниже 0 градусов. В этом случае число сравнений очередного значения температуры с нулем заранее неизвестно, а задано только условие, определяющее число шагов команды повторения.

В программировании и те, и другие циклы встречаются одинаково часто, поэтому в алгоритмических языках существуют две формы записи команд повторения. Одну из них назовем циклом с известным числом шагов (цикл *для*), а другую циклом по условию (цикл *пока*). Последняя форма является универсальной, так как циклы типа *для* легко могут быть представлены через нее. Если K — переменная, фиксирующая число шагов команды повторения, то цикл начинается со значения K, равного единице, а заканчивается со значением K, равным заранее известному числу. В нашей первой задаче с температурами конечное значение K равно 31, т. е. условие окончания работы цикла может быть записано как K>31.

Если цикла типа *пока* на языке Бейсик нет и его нужно строить на основе оператора условного перехода, то цикл типа *для* имеется. Вот формат этого оператора.

```
10 FOR K=KH TO   KK STEP KP FOR—для
20 . . .                      TO — к
   . . .                      STEP — шаг
   . . .                   NEXT — следующий
90 NEXT K
100 . . .
```

Здесь K — переменная, называемая обычно параметром цикла и фиксирующая число повторений группы команд, составляющих тело цикла; KH и KK — начальное и конечное значение параметра цикла; KP — приращение параметра на каждом шаге цикла. Значение KP может быть как положительным, так и отрицательным. Если значение KP равно единице, то конструкцию STEP 1 в записи заголовка (в нашем случае это оператор с номером 10) цикла опускают, он принимает вид

10 FOR K=KH TO KK

Изменение параметра цикла на величину шага происходит в операторе NEXT K. После его выполнения ЭВМ возвращает управление к оператору FOR, в котором происходит сравнение текущего значения параметра с конечным; если оно не больше конечного, то снова выполняется тело цикла, иначе происходит передача управления к оператору, следующему за NEXT. Таким образом, FOR — сложная команда. Встречая ее, машина выполняет целую последовательность более простых команд, которую можно на Бейсике представить так:

```
5   LET K=KH
10  IF K>KK THEN GO TO 100
    . . .
    . . .
    . . .
80  LET K=K+KP
90  GO TO 10
100 . . .
```

Эту последовательность команд часто называют семантикой оператора цикла типа *для*.

Приведем примеры использования команд повторения обоих типов.

1. Циклы для вычисления суммы линейного массива, описанного как DIM X(50).

Цикл типа *пока*

```
. . . . .
100 LET K=0
105 LET C=0
110 IF K>50 THEN GO TO 130
115 LET C=C+X(K)
120 LET K=K+1
125 GO TO 110
130 . . .
```

Цикл типа *для*

```
. . . . .
100 LET C=0
105 FOR K=0 TO 50
110 LET C=C+X(K)
115 NEXT K
```

```
120 . . .
```

2. Циклы для определения номера и значения максимального элемента того же массива.

Цикл типа *пока*

```
. . . .
50  LET K=1
55  LET M=X(0)
60  LET N=0
65  IF K>50 THEN 100
70  IF M<X(K) THEN 80
75  GO TO 90
80  LET M=X(K)
85  LET N=K
90  LET K=K+1
95  GO TO 65
100 . . .
```


Цикл типа *для*

```
50 LET K=1
55 LET M=X(0)
60 LET N=0
65 FOR K=1 TO 50
70 IF M<X(K) THEN 80
75 GO TO 90
80 LET M=X(K)
85 LET N=K
90 NEXT K
95 . . .
```

Обратите внимание на написание команд условного перехода в приведенных фрагментах: после слова THEN в них отсутствует команда GO TO, а сразу указан номер оператора, на который должен быть совершен условный переход. Такая запись допускается в Бейсике.

Полезно сравнить между собой эти фрагменты программ, чтобы уяснить разницу между двумя способами оформления команд повторения. В цикле типа *для* часть действий, которые программист должен явно указывать при оформлении цикла типа *пока* (LET K=1, LET K=0, LET K=K+1, GO TO), «упрятывается» в семантику, выполняется машиной самостоятельно и, следовательно, становится для программиста неявной.

Обратите внимание и на такой момент: в заголовке цикла пока—нц—кц и в операторе условного перехода, играющем роль заголовка в эквивалентной конструкции Бейсика, указываются условия, имеющие противоположные (инверсные) значения истинности. В первом случае в заголовке указывается условие, соблюдение которого определяет выполнение тела цикла, а во втором — условие, определяющее выход из цикла.

Прежде чем перейти к написанию алгоритмов на Бейсике с использованием команд присваивания и эквивалентов команд ветвления и повторения, рассмотрим имеющиеся в нем команды для ввода и вывода значений величин. Как уже отмечалось выше, Бейсик — язык, реализованный на диалоговых вычислительных комплексах, поэтому в нем имеются удобные операции для ввода данных с клавиатуры терминала и вывода результатов на экран. Мы ограничимся рассмотрением только этих команд.

Операторы ввода и вывода

Умел взять — умей и отдать.

При оформлении алгоритмов часть величин относилась к аргументам, часть — к результатам, некоторые являлись промежуточными величинами. Если алгоритм записывается как программа, которая должна выполняться на ЭВМ, значения аргументов должны быть введены в память ЭВМ, значения результатов должны быть введены (напечатаны).

Организовать ввод в Бейсике можно двумя способами.

1. Оператор INPUT (ввод) имеет вид:

INPUT имя переменной, имя переменной...

Когда он встречается в процессе выполнения программы, машина останавливается и ждет ввода значения для имени переменной. На экране в этот момент появляется знак вопроса (?). Требуемое значение программист должен набрать на клавиатуре ЭВМ. Если в операторе INPUT стоит несколько имен переменных, ЭВМ продолжает высвечивать знак вопроса до тех пор, пока не будут введены все требуемые значения. Таким образом, если в прогремме описан линейный массив A в операторе DIM A(4), то для ввода значений его элементов можно написать

INPUT A(0), A(1), A(2), A(3), A(4)

Работа за терминалом будет идти так:

```
? 3.6 ВК
? 8.12 ВК
? 5,4 ВК
? 0.16 ВК
? 9.9 ВК
```

(**ВК** — клавиша, которую надо нажать для перехода на новую строку).

Возможен и такой вариант ввода значений:

```
? 3.6, 8.12, 5.4, 0.16, 9.9 ВК
```

(вводимые значения — через запятую, в записи числа — точка!).

2. Операторы READ (читать) и DATA (данные) используются только совместно! Бейсик позволяет по-разному группировать их, но лучше писать программу так, чтобы каждому оператору READ соответствовал бы свой оператор DATA.

Формат оператора READ:

READ имя аргумента, имя аргумента,...

Формат оператора DATA:

DATA значение, значение,...

И значения, и имена разделяются запятыми; если не оговорено специально другое, количества имен и значений должны быть равны друг другу.

Пример использования операторов READ и DATA для ввода элементов описанного выше массива A:

```
DATA 3.6, 8.12, 5.4, 0.16, 9.9
READ A(0), A(1), A(2), A(3), A(4)
```

В операторе DATA значения «хранятся», а вводятся оператором READ в соответствующие переменные. Поскольку операторы DATA выполняются в программе не самостоятельно, а только вместе с операторами READ, они могут располагаться в ней в произвольных местах. Если имеется несколько DATA—READ, машина сама устанавливает их соответствие в порядке написания, но лучше писать соответствующие друг другу операторы READ и DATA рядом друг с другом.

В чем разница между двумя описанными способами ввода? В случае оператора INPUT значения для переменных вводятся с клавиатуры терминала. Выполнение программы прекращается, и ЭВМ ждет, когда соответствующее значение будет набрано на клавиатуре.

Во втором случае остановка программы не происходит, так как значения вводимых данных находятся в самой программе. ЭВМ находит соответствующий оператор DATA и из него берет необходимые значения. Если в имеющихся операторах DATA не хватает значений для оператора READ, ЭВМ напечатает на экране сообщение об ошибке.

Помните о разных способах записи значений для целых, вещественных и литерных величин. Так, например, в Бейсике ДВК-2 для оператора

INPUT A%, B%, T$

на клавиатуре должны быть набраны два литерных значения и одно целое, например «Я ЗНАЮ», 2 %, «ОПЕРАТОРА ВВОДА»

Остановимся подробнее на организации ввода массивов. Если массив содержит большое число элементов, то неудобно перечислять их имена (переменные с индексами) в операторах ввода. Удобнее воспользоваться оператором цикла.

Ниже приведены фрагменты программы, организующие ввод элементов линейного массива:

```
10 DIM A(5)
20 FOR I=0 TO 5
30 INPUT A(I)
40 NEXT I
```

```
10 DIM A(5)
20 DATA 2,2,4,4,6,6,
30 FOR I=0 TO 5
40 READ A(I)
50 NEXT I
```

Первая программа шесть раз высветит «?» для ввода всех значений элементов массива A, которые надо будет последовательно набрать на клавиатуре. Во второй оператор READ A(I) прочитает эти шесть значений из оператора DATA.

Ввод двухмерных массивов (прямоугольных таблиц) потребует написания вложенных циклов. Ниже приведен пример ввода элементов литерного двухмерного массива A.

```
10 DIM A$(4,2)
20 ROF I=0 TO 4
30 FOR J=0 TO 2
40 INPUT A$(I, J)
50 NEXT J
60 NEXT I
```



```
? « » ВК
?  «МАЛО» ВК
?  «СКАЗАТЬ» ВК
?  «НЕ» ВК
```

и т. д.

```
10 DIM A (4,2)
20 DATA « », «МАЛО», «СКАЗАТЬ»
21 DATA «НЕ», «ДЕЛАЙ», «ТАК.»
22 DATA «НАДО», «ЕЩЕ», «ПРИБАВИТЬ:»
23 DATA «ВОТ», «ТАК», «ДЕЛАЙ.»
24 DATA «А.», «М.» «ГОРЬКИЙ»
25 FOR I=0 TO 4
30 FOR J=0 TO 2
35 READ A (I, J)
40 NEXT J
45 NEXT I
```

Для первой показано начало ввода. Двухмерный массив заполняется построчно. Во второй программе — несколько операторов DATA. Значения массива A будут выбираться из них в порядке возрастания номеров операторов.

Вывод в Бейсике организуется так:

PRINT имя переменной ⟨разделитель⟩ имя переменной ⟨разделитель⟩

Здесь разделитель — либо запятая, либо точка с запятой, например:

```
PRINT A, B, C; B2, M
```

Что означают разделители?

На экране терминала обычно 24 строки, на которых располагаются выводимые значения. В каждой строке 5 зон, содержащих по 14 позиций. В одной позиции может быть записан один символ (буква, цифра,

знак). После заполнения одной строки ЭВМ автоматически переходит на следующую. Разделители могут изменить этот порядок.

Если разделитель — запятая, то следующий элемент вывода печатается в следующей зоне; если точка с запятой, то производится упакованная печать, т. е. следующий элемент вывода печатается, начиная с первой свободной позиции в текущей строке (на различных ДВК могут быть разные варианты упакованной печати). Если стоят две запятые подряд („), то при печати пропускается одна зона.

Помните: литерные данные печатаются без ограничительных кавычек; вещественное число печатается в форме с фиксированной точкой, если оно из диапазона $0.01 \div \div 999999$, в остальных случаях в форме с плавающей точкой; оператор PRINT без списка имен вывода выводит строку пробелов, иными словами, осуществляет переход к следующей строке.

В Бейсике имеются дополнительные возможности управления выводом (например, печать графика). Мы не будем рассматривать их; приведем только для примера фрагмент программы, обеспечивающий печать двухмерного вещественного массива в виде соответствующей таблицы.

```
10  DIM A(3,2)
          . . .
          . . .
          . . .
100 FOR I=0 TO 3
110 FOR J=0 TO 2
120 PRINT A(I, J);
130 NEXT J
140 PRINT
150 NEXT I
. . .
```

Оператор 120 обеспечит упакованную печать строки массива, оператор 140 обеспечит переход к следующей строке на дисплее.

Встроенные функции языка Бейсик

При расчетах часто бывает необходимо вычислить значение квадратного корня из некоторого числа, логарифма или одной из тригонометрических функций. В Бейсике есть набор так называемых встроенных (или стандартных) функций специально для вычисления подобных вещей (см. табл.).

Некоторые варианты Бейсика имеют более широкий набор стандартных функций, включающий перевод из радианной меры в градусную, десятичный логарифм, arcsin и arccos, гиперболические функции и т. д. Программист должен помнить, что прежде чем пользоваться некоторой стандартной функцией, он обязательно должен прочитать описание ее реализации в конкретном ДВК.

Рассмотрим подробнее стандартную функцию, вырабатывающую псевдослучайное число с равномерным распределением в интервале (0,1) RND(X) (сокращение от random, что означает — «случайный»). Обозначение RND называется указателем функции. Для получения случайного числа можно написать оператор

```
LET E=RND(X)
```

В большинстве версий Бейсика при его выполнении переменная E получит некоторое случайное значение, находящееся в интервале от 0 до 1. Эти значения непредсказуемы, каждое из них может появиться с такой же вероятностью, как и другое, поэтому они и называются равномерно распределенными в интервале (0,1).

Обозначение встроенной функции	Назначение функции	Аргумент	Результат
SQR(X)	$\sqrt{x}$	X>0	вещественный
SIN(X)	$\sin x$	X в радианах	»
COS(X)	$\cos x$	»	»
TAN(X)	$\operatorname{tg} x$	»	»
ATN(X)	$\operatorname{arctg} x$	X — вещественное	в радианах
LOG(X)	$\ln x$	X>0	вещественный
EXP(X)	e^x	X — вещественное	»
ABS(X)	$\lvert x \rvert$	»	»
INT(X)	ent x	»	наибольшее целое, не превосходящее X
SGN(X)	$\operatorname{sign} x = \begin{cases} 1, & x>0 \\ 0, & x=0 \\ -1, & x<0 \end{cases}$	»	1 при X>0 0 при X=0 −1 при X<0
RND(X)	генерирует псевдослучайное число с равномерным распределением в (0,1)		вещественный

Для получения и печати нескольких случайных чисел необходимо воспользоваться оператором цикла:

```
10 FOR K=1 TO 10
20 LET E=RND(X)
30 PRINT E
40 NEXT K
50 STOP
```

Несколько слов об аргументе X этой функции. Может показаться странным, но аргументу X не надо присваивать никакого значения перед обращением к функции RND. Попросту говоря, он не нужен, но (!) писать его надо — таковы общие требования, предъявляемые к стандартным функциям в Бейсике.

И еще одно важное замечание. Если к приведенной выше программе обратиться несколько раз, то она несколько раз выдаст одну и ту же последовательность случайных чисел. Для получения разных последовательностей надо в начале программы поместить специальный оператор RANDOMIZE:

```
5  RANDOMIZE
10 FOR K=1 TO 10
20 LET E=RND (X)
30 PRINT E
40 NEXT K \ STOP
```

В. РОМАНОВ

Информатика и прикладная математика

Программа для школ (классов) с углубленным изучением математики

Программа курса информатики и прикладной математики для школ (классов) с углубленным теоретическим и практическим изучением математики охватывает весь материал, содержащийся в курсе информатики средней общеобразовательной школы. При этом имеется в виду, что учащиеся должны овладеть знаниями, умениями и навыками по программе этого курса и, сверх того, знаниями, умениями и навыками, указанными ниже.

Курс рассчитан на изучение в VII—X классах за счет часов, отводимых на трудовое обучение согласно типовому учебному плану для школ указанного типа, а также на уроках информатики в IX и X классах.

Программа составлена на основе опыта многолетнего преподавания соответствующего экспериментального курса в школе № 444 г. Москвы и изучения опыта других математических школ.

Курс должен не только более широко решать задачи, поставленные при изучении информатики в средней общеобразовательной школе; одна из основных его целей — практически увязать изучение математики с реальной действительностью, научить применять математические методы при изучении законов окружающего мира и для решения задач, возникающих на практике.

Большинство современных прикладных задач могут быть решены только с помощью быстродействующих ЭВМ, и поэтому другой целью курса является обучение программированию, позволяющему записать алгоритм решения на изучаемом языке, провести отладку программы и счет на ЭВМ и выполнить анализ результатов.

Курс состоит из трех разделов.

1. «Основы алгоритмизации и программирования».

2. «Вычислительная техника: принципы устройства и сферы применения».

3. «Методы прикладной математики и построение математических моделей при решении задач на ЭВМ».

Программой предусматривается параллельное изучение этих разделов. Это позволяет в разделе «Основы алгоритмизации и программирования» составлять программы содержательных математических задач. Курс прикладной математики (а также и другие школьные курсы) получает богатый иллюстративный материал для использования на уроках. Помимо этого, параллельное изучение разделов курса позволяет при необходимости привлечь к занятиям преподавателей — специалистов в областях программирования, вычислительной техники и прикладной математики.

Первый раздел курса «Основы алгоритмизации и программирования» построен как последовательность тщательно подобранных задач постепенно возрастающей сложности, начиная от простейших (дидактического характера) и кончая содержательными задачами прикладной математики. Сначала рассматривается построение алгоритма решения задачи и лишь затем его реализация на конкретном языке программирования. Таким образом, те или иные конструкции, технические средства языка, а также различные приемы программирования вводятся по мере возникновения в них необходимости в процессе решения задачи. При изучении материала первого раздела предусматривается запись каждого алгоритма на псевдокоде или алгоритмическом языке и реализация его на одном из языков программирования, который выбирается школой в зависимости от имеющихся возможностей.

При отсутствии или ограниченном доступе к работе в дисплейном классе реализация изучаемых алгоритмов может производиться в пакетном режиме или с помощью программируемых микрокалькуляторов. В этом случае работу с текстовой и графической информацией следует организовать в период летней трудовой практики в вычислительном центре базового предприятия, институте усовершенствования учителей, педагогическом институте и т. д. при содействии органов народного образования. Возможно также комбинированное использование вычислительной техники.

В теме 1.1 учащиеся знакомятся с разными типами алгоритмов и их реализацией различными формальными исполнителями. Рассматриваются разные способы представления алгоритмов, в том числе условные графические схемы (блок-схемы) и таблицы решений.

Изучение темы 1.2 наиболее эффективно при использовании программируемого микрокалькулятора, который позволяет наглядно продемонстрировать основные принципы автоматической обработки информации на ЭВМ, в частности, взаимодействие счетчика и регистра команд в процессе исполнения программы.

Помимо упомянутых основных принципов учащиеся должны знать:

понятие ячейки памяти и ее свойства;

понятие команды, ее адреса и составные части команды;

свойства счетчика и регистра команд и их роль в процессе исполнения программы, а также работу команды безусловного перехода.

Учащиеся должны уметь составить, записать в память и исполнить линейную программу в однотактном и автоматическом режимах.

В результате изучения темы 1.3 учащиеся, помимо понятий, указанных в программе, должны знать правила записи линейных алгоритмов на языках алгоритмических и программирования, иметь представление о трансляции и интерпретации, уметь исполнять линейные алгоритмы на персональной ЭВМ в режиме непосредственных вычислений и программирования, уметь пользоваться текстовым и графическим редактором (при наличии соответствующего программного обеспечения).

Изучение темы 1.4 на материале линейных алгоритмов позволит использовать вспомогательные алгоритмы и подпрограммы при решении задач практически с самого начала курса. При этом первое время достаточно ограничиться вспомогательными алгоритмами вычисления значений функций (подпрограммами-функциями). Стандартные функции вводятся как один из видов вспомогательных алгоритмов.

По окончании изучения этого материала учащиеся должны знать правила записи алгоритма и обращения к вспомогательным алгоритмам и подпрограммам-функциям, уметь использовать их в необходимых случаях. На первоначальном этапе рассматривать вложенные вспомогательные алгоритмы нет необходимости.

При использовании в качестве исполнителя алгоритмов программируемого микрокалькулятора в данной теме изучается работа команды безусловного перехода с возвратом и ее взаимодействие с регистром возврата, а в дальнейшем, при составлении вложенных подпрограмм, со всем стеком возврата.

При изучении ветвлений рассматриваются как следование, так и вложение ветвлений; используются составные условия. Следует иметь в виду, что при реализации ветвлений на языке программирования на долю условных переходов выпадает значительное число ошибок логического характера, допускаемых учащимися. Их количество может быть уменьшено, если при составлении программы на языке программирования соблюдать правила структурирования программы.

Понятие цикла может быть, естественно, введено на примере вычисления функции от аргумента, изменяющегося в заданных границах с постоянным шагом. Далее это понятие получает развитие на материале числовых последовательностей, заданных как формулой общего члена с проверкой окончания при помощи естественного счетчика — номера члена последовательности, так и рекуррентно, когда возникает необходимость в специальном счетчике циклов, вводимом искусственно. Дальнейшее усложнение состоит в суммировании или перемножении членов последовательностей при вычислении конечных сумм и произведений.

Первыми примерами задач на вложенные циклы могут быть табулирование функций двух и более аргументов, вычисление двойных сумм и произведений и т. д.

В теме раздела «Циклы с разветвлениями внутри тела цикла» рассматриваются и вложенные циклы с разветвлениями. Представление об уровне задач дает следующий перечень: алгоритм Евклида, функция Эйлера, треугольник Паскаля, решето Эратосфена, задача о «счастливых» билетах, табулирование функции, заданной в виде ряда, и т. д.

Из задач, где проверка окончания цикла производится по достигнутой точности вычислений (итерационные циклы), можно рассмотреть алгоритм извлечения квадратного корня методом Ньютона, вычисление «золотого сечения» как предела последовательности чисел Фибоначчи, вычисление числа π как предела полупериметров вписанных многоугольников, числа e как предела последовательности $(1+\frac{1}{n})^n$. В дальнейшем эти алгоритмы широко используются при изучении вычислительной математики: итерационные методы решения уравнений и систем уравнений, численное интегрирование, вычисление элементарных функций с помощью рядов, простейшие численные методы решения дифференциальных уравнений — все они реализуются с помощью итерационных циклов.

При решении задач следует широко применять метод последовательной детализации алгоритмов, прибегая к использованию вспомогательных алгоритмов и подпрограмм. Так, в задаче табулирования функции целесообразно оформить в виде подпрограммы вычисление значения функции, в задачах на вложенные циклы удобно выделять в виде вспомогательного алгоритма внутренний цикл. При использовании технологии «сверху вниз» основная задача программирования может быть представлена как реализация одного исполнителя на базе другого.

Первыми задачами темы «Алгоритмы обработки массивов информации» могут быть алгоритмы поэлементной обработки массива — выполнить над каждым элементом массива совокупность некоторых операций (например, возвести каждый элемент числового массива в квадрат). Следующая задача — суммирование или перемножение элементов первоначального и преобразованного массива (например, сложить квадраты всех элементов). К выполнению операций над двумя массивами сводится вычисление скалярного произведения $\sum_{i=1}^{n} a_i b_i$. Порядок усложнения задач в принципе такой же, как и в предыдущем разделе: рассматриваются вложенные циклы (например, типа $\sum_{i=1}^{n}\prod_{j=1}^{m} a_i b_j$; циклы с разветвлениями внутри тела цикла (например, поиск элементов сначала по одному, а затем и нескольким свойствам, при этом элементами массива могут быть как числа, так и другие объекты); вложенные циклы с разветвлениями внутри тела цикла (сортировка массива, построение гистограммы, поиск локальных экстремумов, минимакса и т. д.).

Следующим типом задач являются задачи на двумерные массивы: традиционные матричные (транспонирование матрицы, нахождение следа матрицы, перемножение матриц и т. д.); преобразование прямоугольной матрицы в квадратную, приведение матриц с вычислением числа приведения и т. д. Прикладной смысл рассматриваемых задач выяснится в дальнейшем при изучении соответствующих тем третьего раздела курса, где будут выполняться расчеты по составленным программам. Например, из двух последних задач одна будет использована при реализации метода наименьших квадратов, другая — в методах ветвей и границ.

В итоге изучения двух последних тем — «Циклы» и «Алгоритмы обработки массивов информации» — учащиеся должны уметь для всех перечисленных задач составить алгоритм решения, используя конструкции «для» и «пока», и программу на языке программирования, используя соответствующие операторы цикла. Несколько задач по выбору учителя должны быть каждым учеником отлажены и реализованы на ЭВМ. При наличии соответствующего программного обеспечения среди этих задач должны быть задачи обработки текстовой и графической информации.

При работе с программируемым микрокалькулятором учащиеся должны уметь программировать циклические задачи, используя как команды условного перехода, так и команды организации цикла. Программируемый микрокалькулятор позволяет понять процессы, происходящие при обработке массивов информации на машинном уровне. Учащиеся должны уметь изменять и восстанавливать переменные команды (в режиме косвенной адресации). Так как память программируемого микрокалькулятора мала, то с его помощью целесообразно решать такие задачи, которые допускают ввод элементов массива по одному, а также обработку небольших массивов информации.

В теме 1.8 раскрываются не изученные ранее дополнительные средства языка программирования, расширяющие возможности пользователя и позволяющие составлять более рациональные программы: вводятся логические выражения и расширяется набор типов величин; раскрываются дополнительные возможности внешних устройств. На примерах конкретных задач демонстрируются специальные приемы программирования: логическая шкала, «мерцающая единица» и т. п. Этой темой завершается изучение языка программирования. В итоге учащиеся должны знать все изученные средства данного языка и уметь применять их при решении задач. Несколько из них по этой теме по указанию учителя должны быть полностью отлажены каждым учеником во время практических занятий.

Тема 1.9 (дополнительная, изучается по желанию учащихся) — заключительная тема первого раздела. Здесь выясняется необходимость уточнения введенного в начале курса интуитивного понятия алгоритма, в ознакомительном плане рассматривается основная гипотеза теории алгоритмов.

Раздел «Вычислительная техника: принципы устройства и сферы применения» начинается с темы «Системы счисления». Учащиеся должны знать принципы записи чисел в позиционных системах счисления, способы перевода чисел из одной системы в другую, причины использования в ЭВМ двоичной системы счисления; уметь выполнять действия в других системах счисления; переводить двоичные числа в шестнадцатеричную систему и обратно.

В теме 2.2 учащиеся должны знать определение функций алгебры логики, названия и обозначения некоторых из них, используемых в дальнейшем; уметь задавать в виде таблицы функции, заданные формулой алгебры логики, и обратно; выражать заданную функцию через базовые и строить функциональные схемы (полусумматора, сумматора, дешифратора и др.); уметь пользоваться полигоном логических схем.

В теме 2.3 учащиеся знакомятся с ЭВМ как комплексом так называемого «жесткого» и «мягкого» оборудования. Дается представление о принципах устройства и работы микропроцессорной техники. Рассматривается процессор микро-ЭВМ (микропроцессор, арифметико-логическое устройство, оперативное запоминающее устройство, постоянное и программируемое, постоянное запоминающее устройство), магистраль, порты ввода и вывода, контроллер, периферийные устройства (магнитные ленты, диски, принтер, джойстик и т. д.). Дается понятие о микропрограммировании.

Далее учащиеся знакомятся с назначением основных систем базового программного обеспечения. Они должны получить на уровне, необходимом пользователю, понятие об операционной системе, системе программирования; об общей схеме функционирования вычислительной системы, об этапах прохождения программы пользователя. Рассматриваются функции операционной системы, сообщается о пакетном, мультипрограммном режимах, о работе в режиме реального времени; изучается работа с языком управления заданиями.

Излагаются основные принципы построения конкретной файловой системы, как способа хранения информации на машинном носителе. Учащиеся знакомятся со структурой файлов, способами их создания, обновления, копирования; каталогом файлов; операциями с ними — поиском, уничтожением, переименованием; управлением файлами на уровне языка управления заданиями; редактированием.

В ознакомительном плане рассматриваются принципы построения транслятора; выделяются основные этапы его работы — лексический и синтаксический анализ, генерация, оформление и выдача объектной программы. Освещаются возможности отладки, диагностика некоторых ошибок, а также способы предотвращения ошибок в программах, вызываемых действием побочных эффектов.

Сообщаются необходимые сведения об имеющемся программном обеспечении; учащиеся знакомятся с типом и составом пакетов прикладных

программ, библиотеками программ.

С рядом методов обработки текстовой и графической информации учащиеся ознакомились в ходе изучения раздела «Основы алгоритмизации и программирования». В данном разделе отрабатываются технические приемы, допускаемые имеющимся программным обеспечением.

Учащиеся должны знать:

назначение и состав базового программного обеспечения;

язык управления заданиями на уровне, необходимом для выполнения школьных программ, типовые сообщения системы об ошибках;

последовательность управления вычислительным процессом;

средства редактирования текстов и программ.

Учащиеся должны уметь:

пользоваться конкретным базовым и прикладным программным обеспечением;

составлять на языке управления заданиями конкретное задание для проведения отладки и счета на ЭВМ;

уметь исправлять обнаруженные ошибки на этапе синтаксического и семантического анализа.

В школах, оснащенных персональными ЭВМ, необходимо сообщить о составе и назначении базового программного обеспечения больших ЭВМ.

Одной из важнейших задач является организация хранения информации, которая наилучшим образом отражала бы связи, существующие в структурах данных реальных задач. Поясняется, что специальные способы организации данных позволяют на их базе решать совокупность задач определенного класса.

Как с фундаментальными структурами данных, так и с динамическими структурами учащиеся знакомятся в процессе решения задач. Например, при решении задач поиска и сортировки используются такие структуры, как стек, очередь, дерево и др. Интерес школьников вызывают задачи типа «О восьми ферзях», «Ханойские башни», «Поиск пути в лабиринте», «Игра в 15» и т. п.

Приводятся сведения об информационно-поисковых системах, базах данных. Дается понятие об электронных таблицах и основных операциях с ними.

На простых примерах раскрываются идеи помехоустойчивого кодирования информации: проверка на четность, коды Хемминга, обнаруживающие и исправляющие одиночные ошибки.

Учащиеся должны знать:

алгоритмы задач поиска и сортировки;

изученные информационные структуры данных.

Учащиеся должны уметь:

составлять программы поиска, включения, исключения и замены элементов;

обращаться к имеющимся базам данных с запросами.

Знакомство с темой 2.5 желательно связать с общественно полезным производительным трудом и трудовой практикой: работой на станках с программным управлением, автоматизированных рабочих местах; участием в создании и поддержании банка данных; изучение и работа с конкретной автоматизированной системой управления и т. д.

В разделе «Прикладная математика» основным является понятие математической модели. Оно дается в развитии и вводится на простых примерах, с которыми учащиеся ранее неоднократно встречались.

Ставится задача формировать способность видеть в исследуемых объектах и процессах наиболее существенные закономерности и описывать их на языке математических понятий, умение пользоваться изученным математическим аппаратом для решения сформулированной задачи и анализировать полученные результаты с точки зрения соответствия их фактическому протеканию исследуемого процесса или состоянию объекта.

Во многих темах раздела «Прикладная математика» углубляется и расширяется материал, изучаемый на уроках математики, а также проводится практическая реализация на ЭВМ рассматриваемых математических методов. Например, на уроках математики изучаются основы теории вероятностей, а в курсе прикладной математики — статистическое моделирование с решением задач методом Монте-Карло. После знакомства на уроках математики с дифференциальными уравнениями, как математическими моделями реальных процессов, в курсе прикладной математики изучаются численные методы решения дифференциальных уравнений. Эти темы должны изучаться в тесной взаимосвязи. Широко следует рассматривать прикладные задачи, приводящие к необходимости использования того или иного метода.

При изучении темы «Источники ошибок и приближенные вычисления» следует подчеркнуть, что точность реализации выбранной модели должна быть оправдана постановкой задачи. При вычислениях необходимо проводить анализ возможных ошибок, которые могут оказаться соизмеримыми с точностью модели. Рассматриваются три основных источника ошибок: ошибки в исходных данных; связанные с выбором математического метода; округления.

Тема начинается в VII классе, где изучаются действия над приближенными числами, и в дальнейшем развивается при изучении каждого численного метода.

Классическим методам вычислительной математики в курсе отводится значительное место. К началу изучения вычислительной математики объем знаний учащихся по программированию позволяет каждую тему завершать составлением программ на языке программирования. Рассматривается применение изучаемого материала к решению прикладных задач физики, химии, биологии, техники.

При решении уравнений учащиеся впервые встречаются с приближенными методами, и важно разъяснить, что такие методы с точки зрения практических приложений не уступают точным.

Системы уравнений решаются как приближенными, так и точными методами Гаусса и Крамера.

Приводятся необходимые сведения о таблицах функций; рассматриваются табличные разности различных порядков и устанавливается их связь с производными; рассматривается задача параболической интерполяции.

Численные методы интегрирования (прямоугольников, трапеций и парабол) изучаются после знакомства на уроках математики с понятием интеграла и формулой Ньютона—Лейбница.

При изучении численных методов решения дифференциальных уравнений основное внимание уделяется методу Эйлера и его модификациям. Дается понятие о моделировании дифференциальных уравнений на аналоговых ЭВМ, принципах решения задач с использованием гибридных ЭВМ.

В теме «Вычисление элементарных функций с помощью рядов» обращается внимание на оценку точности получаемых значений функций, скорость сходимости исследуемого ряда.

По окончании изучения методов вычислительной математики учащиеся должны:

знать действия с приближенными числами и уметь выполнять вычисления с учетом погрешностей арифметических действий;

знать методы и геометрическую интерпретацию решения уравнений и уметь решать уравнения методами половинного деления, хорд и касательных, итераций, проводя оценку точности решения в доступных случаях;

знать свойства определителей и способы их вычисления, начальные сведения из алгебры матриц, критерий совместимости системы линейных уравнений, уметь исследовать и решать системы линейных уравнений методами Гаусса и Крамера, а также итерационным методом;

знать свойства конечных разностей и уметь строить интерполяционный многочлен Ньютона или Лагранжа;

знать разложения в ряды основных элементарных функций и уметь с помощью вычислительной техники вычислять суммы сходящихся рядов, проводя в доступных случаях оценку точности;

знать понятия общего и частных решений, а также геометрический смысл решений дифференциального уравнения;

уметь вычислять площади фигур методами прямоугольников, трапеций, парабол; решать линейные дифференциальные уравнения усовершенствованным методом Эйлера, а также пользоваться готовыми формулами Рунге-Кутта.

Во время обучения у учащихся формируется представление о явлениях окружающего мира как о детерминированных процессах. При изучении темы «Статистическое моделирование» учащиеся встречаются с более общими закономерностями особого рода, когда при построении математической модели требуется учитывать случайные факторы. Тогда прибегают к универсальному методу статистического моделирования. Идея метода проста и доступна учащимся (пояснить ее удобно на примерах задачи типа Бюффона).

Применение метода демонстрируется при решении прикладных задач расчета систем массового обслуживания, расчета качества и надежности изделий. При выполнении подобных работ учащиеся получают опыт статистической обработки результатов машинного эксперимента.

После изучения темы учащиеся должны знать:

понятие дискретных и непрерывных случайных величин, их свойства, некоторые типы распределений;

общую схему применения метода Монте-Карло;

методы получения случайных величин на ЭВМ и их преобразование;

принцип и схему расчета системы массового обслуживания, расчета качества и надежности изделий.

Учащиеся должны уметь:

пользоваться датчиком случайных величин и преобразовывать их;

применять метод Монте-Карло для решения четко поставленной задачи.

При изучении методов обработки результатов измерений необходимо исключить влияние грубых, а также систематических ошибок. Знакомство со случайными числами, нормальным законом распределения позволяет рассмотреть вопрос о свойствах случайных ошибок и учете их влияния на результаты измерений.

При нахождении параметров эмпирических зависимостей применяется метод наименьших квадратов. На лабораторных занятиях выполняется аппроксимация функции полиномом заданной степени.

Учащиеся должны знать классификацию ошибок измерений, их свойства; уметь обрабатывать на ЭВМ результаты измерений по способу наименьших квадратов.

Линейные зависимости в задачах об организации работы транспорта, планировании ресурсов производства, распределении ресурсов делают их доступными для решения школьниками. При решении основной задачи линейного программирования применяется весьма естественный для понимания симплекс-метод; особо выделяется транспортная задача.

Первая часть темы «Элементы линейной алгебры и линейного программирования» изучена в курсе вычислительной математики. В результате изучения линейного программирования учащиеся должны знать:

примеры задач линейного программирования различных типов;

геометрический смысл задачи линейного программирования и графический метод ее решения;

алгоритм симплекс-метода.

Учащиеся должны уметь:

применять алгоритм решения транспортной задачи в конкретных ситуациях.

Решение оптимизационных комбинированных задач проводится с помощью целенаправленного перебора (выясняется практическая бесперспективность полного перебора вариантов ввиду их огромного количества при решении даже простых задач). Основными способами решения являются методы ветвей и границ. Демонстрируется идея случайного поиска.

Для выявления свойств связанных между собой объектов используются графы; они, в частности, применяются при решении задач сетевого планирования и управления. Учащиеся знакомятся с приложением сетевых моделей к решению комбинаторных задач. Школьникам указывается на возможность применения вероятностных методов в сетевом планировании.

По теме «Оптимизационные задачи дискретной математики» учащиеся должны знать:

постановку задач коммивояжера, о рюкзаке, назначениях, двух и трех станков, и т. п.;

идею метода последовательного отсеивания вариантов;

идею методов ветвей и границ;

понятие сетевого графика.

Учащиеся должны уметь:

строить дерево вариантов применительно к рассматриваемой задаче;

сводить одну оптимизационную задачу к другой;

составлять алгоритм и программы решения задачи типа «Расстановка восьми ферзей»;

выполнять построение сетевого графика для учебной задачи и определять критический путь.

В IX и X классах предусмотрено время для изучения темы по выбору школы или базового предприятия. Обычно такая тема обусловлена спецификой базового предприятия и, как правило, связывается с общественно полезным производительным трудом и трудовой практикой. Школа может использовать указанное время на углубление какой-либо программной или изучение новой темы. Время, отведенное на летнюю трудовую практику, используется также для реализации на ЭВМ задач, алгоритмы решения которых были разобраны на уроках.

В последнем учебном году учащиеся работают над прикладной задачей значительного объема, как правило производственной, под руководством специалистов базового или других предприятий. В необходимых случаях эта работа проводится за счет времени, отводимого на производительный труд. По окончании работы составляется отчет, который обычно содержит такие разделы: описание проблемы, постановка задачи, выбор и обоснование метода решения, построение математической модели, разработка программы, особенности реализации ее на ЭВМ, анализ полученных результатов.

Техническое и программное обеспечение курса непрерывно развивается, что предоставляет все новые возможности для его дальнейшего совершенствования и обогащения.

Содержание обучения

Введение.

Информация, ее виды и свойства. Хранение, преобразование, передача информации. Единицы измерения информации.

1. Основы алгоритмизации и программирования.

1.1. Понятие об алгоритмах.

Примеры алгоритмов. Исполнители алгоритмов. Система предписаний исполнителя и исполнение алгоритмов. Способы представления алгоритмов. Свойства алгоритмов.

Алгоритмы управления роботом (с выводом графической информации): линейные, разветвляющиеся, циклические. Вспомогательные алгоритмы.

1.2. ЭВМ как исполнитель алгоритмов.

Основные принципы автоматической обработки информации на ЭВМ: принцип программного управления, принципы адресности, хранимой программы. Программа как форма представления алгоритма. Память ЭВМ и ее свойства, процессор и его работа, устройства ввода-вывода. Первое понятие об этапах решения задачи на ЭВМ. Программируемый микрокалькулятор как модель простейшей ЭВМ.

1.3. Линейные алгоритмы работы с величинами и начальные сведения о языке программирования. Типы данных.

Величины числовые и символьные (литерные).

Данные: исходные (аргументы), промежуточные, результаты.

Алфавит языка программирования. Переменные и константы.

Ввод и вывод информации. Формы представления информации (числовой, графической, символьной). Арифметические выражения. Оператор присваивания.

Работа ЭВМ в режиме непосредственных вычислений и в режиме программирования. Понятие о трансляции.

Работа с графическим и текстовым редакторами.

1.4. Составление алгоритмов и программ методом пошаговой детализации.

Постановка задач. Вспомогательные алгоритмы. Запись вспомогательных алгоритмов на языке программирования. Подпрограммы. Функции, определяемые пользователем. Способы обращения к подпрограммам. Организация программы (на примере линейной программы). Задачи с использованием графической и текстовой информации. Элементы технологии программирования.

Реализация вспомогательных алгоритмов на программируемом микрокалькуляторе: команда безусловного перехода с возвратом.

1.5. Ветвления.

Постановка задачи. Полные и неполные ветвления. Выбор.

Запись ветвлений на языке программирования.

Ветвления в задачах текстовой и графической информации. Реализация ветвлений на программируемом микрокалькуляторе: команды условного перехода.

1.6. Циклы.

Постановка задачи. Структура цикла. Циклы с пост- и предусловием. Циклы «для» и «пока».

Запись цикла на языке программирования.

Циклы с разветвлением внутри тела цикла.

Циклы с проверкой окончания по достигнутой точности вычислений (итерационные).

Вложенные циклы.

Циклы в задачах с использованием текстовой и графической информации.

Организация цикла на программируемом микрокалькуляторе с помощью команд условного перехода и команд цикла.

1.7. Алгоритмы обработки массивов информации.

Постановка задачи. Таблицы как способ организации данных.

Запись алгоритмов обработки информации на языке программирования. Массивы как структуры данных. Способы описания массивов. Индексные переменные.

Циклы с разветвлением внутри тела цикла.

Вложенные циклы.

Принципы организации обработки массивов информации на программируемом микрокалькуляторе. Косвенная адресация.

1.8. Дополнительные возможности изучаемого языка программирования. Методы и приемы программирования на этом языке.

1.9. Заключение. Дальнейшее развитие понятия алгоритма.

2. Вычислительная техника: принципы устройства и сферы применения.

2.1. Системы счисления.

Непозиционные и позиционные системы счисления. Способы перехода чисел из одной системы в другую. Связь между системами счисления с основанием 2^k. Понятие об экономичности системы счисления.

2.2. Функции алгебры логики и построение функциональных схем.

Двоичные функции от двоичных аргументов. Выражение функций алгебры логики через базовые. Основные законы алгебры логики. Построение функциональных схем полусумматора, сумматора, дешифратора и др.

2.3. Устройство и математическое обеспечение ЭВМ.

Сведения о микроэлектронной и микропроцессорной технике. Основные устройства ЭВМ. Процессор. Интерфейс. Контроллер. Периферийные устройства. Понятие о микропрограммировании.

Организация исполнения программы на ЭВМ. Базовое программное обеспечение. Операционная система. Язык управления заданиями. Организация наборов данных.

Основные принципы построения конкретной файловой системы. Структуры файлов. Управление файлами.

Прикладное программное обеспечение. Типы пакетов: библиотека программ и проблемно-ориентированные системы.

Язык запросов. Документирование программ.

2.4. ЭВМ как система хранения, поиска, передачи и обработки данных.

Методы хранения и поиска информации. Информационно-поисковые системы и базы данных.

Структуры данных.

Понятие о помехоустойчивом кодировании.

2.5. ЭВМ как управляющая система.

Автоматизация управления технологическими процессами и устройствами. Робототехника. Новая информационная технология. Автоматизированные рабочие места. Автоматизированные системы управления. Системы автоматического проектирования.

3. Методы прикладной математики и построение математических моделей при решении задач на ЭВМ.

3.1. Решение задач на ЭВМ.

Математическая постановка задач. Технология алгоритмизации и программирования. Понятие правильности алгоритма. Систематическое конструирование алгоритмов. Корректность программ. Виды формальных моделей. Понятие математической модели.

3.2. Методы вычислительной математики.

Источники ошибок и приближенные вычисления. Решение линейных уравнений методом Гаусса. Итерационные методы решения уравнений и систем. Табличные разности. Интерполяция. Численное интегрирование. Численные методы решения дифференциальных уравнений. Понятие о моделировании дифференциальных уравнений на аналоговых ЭВМ. Представление о решении оптимизационных задач непрерывной математики.

3.3. Элементы линейной алгебры и линейного программирования.

Матрицы и определители. Постановка задачи линейного программирования. Симплекс-метод. Транспортная задача.

3.4. Оптимизационные задачи дискретной математики.

Графы и задачи сетевого планирования и управления. Методы ветвей и границ. Задача о рюкзаке. Задача о назначениях. Задачи двух и трех станков. Задача коммивояжера. Решение комбинаторных задач методом случайного поиска.

3.5. Метод статистического моделирования.

Дискретные и непрерывные случайные величины и их характеристики. Получение на ЭВМ случайных чисел и их преобразование. Типы распределений случайных величин. Моделирование систем массового обслуживания. Способы повышения надежности.

3.6. Обработка результатов измерений.

Ошибки измерений грубые, систематические, случайные. Метод наименьших квадратов.

4. Тема по выбору школы или базового предприятия.

5. Выполнение отчетной работы.

Постановка задачи. Выделение предметной области, классификация объектов и их свойств. Построение математической модели. Построение алгоритма и разработка программы. Реализация программы на ЭВМ. Анализ результатов. Уточнение модели.

Тематическое планирование учебного материала

VII КЛАСС (3 ч в неделю, всего 102 ч)

Основы производства. Выбор профессии.	51 ч
1. Введение	4 ч
2. Понятие об алгоритмах	12 ч
3. Приближенные вычисления (с применением микрокалькулятора)	15 ч
4. Системы счисления	10 ч
5. Функции алгебры логики и построение функциональных схем	10 ч

VIII КЛАСС (3 часа в неделю, всего 102 ч)

Основы производства. Выбор профессии.	17 ч
1. Основные принципы автоматической обработки информации на ЭВМ	12 ч
2. Линейные алгоритмы работы с величинами и начальные сведения о языке программирования	15 ч
3. Составление алгоритмов и программ методом пошаговой детализации	8 ч
4. Ветвления	6 ч
5. Циклы	14 ч
6. Алгоритмы обработки массивов информации	30 ч

IX КЛАСС (5 ч в неделю, всего 170 ч)

1. Дополнительные возможности изучаемого языка программирования. Методы и приемы программирования на этом языке	34 ч

2. Устройство и математическое обеспечение ЭВМ 24 ч
3. ЭВМ как система хранения, поиска, передачи и обработки данных 34 ч
4. ЭВМ как управляющая система 10 ч
5. Методы вычислительной математики 34 ч
6. Тема по выбору школы или базового предприятия 34 ч

X КЛАСС (6 ч в неделю, всего 204 ч)

1. Решение задач на ЭВМ 10 ч
2. Методы вычислительной математики (продолжение) 24 ч
3. Элементы линейной алгебры и линейного программирования 34 ч
4. Оптимизационные задачи дискретной математики 34 ч
5. Метод статистического моделирования 24 ч
6. Обработка результатов измерений 10 ч
7. Тема по выбору школы или базового предприятия 68 ч

Литература

VII класс

1. *Р. Гутер, Полунов Ю.* От абака до компьютера. М., 1981.
2. *Г. Звенигородский.* Первые уроки программирования. М., 1985.
3. Избранные вопросы математики. VII—VIII классы. Под редакцией В. Фирсова. М., 1978.
4. *Н. Розов, Б. Сендов.* Предыстория рождения компьютера // Квант. 1985. № 9, 10.

VIII класс

1. *С. Абрамов.* Элементы программирования. М., 1982.
2. *И. Антипов.* Программирование. М., 1976.
3. *Н. Демидович, В. Монахов.* Программирование на ЭВМ. М., 1977.
4. *В. Каймин.* Решение задач и построение алгоритмов // Квант, 1986, № 10.
5. *И. Клумова.* Игра «Жизнь» // Квант, 1974, № 9.
6. *А. Салтовский, Ю. Первин.* Как работает ЭВМ? М., 1986.
7. *Л. Штернберг.* Программирование на ПМК // Квант. 1986, № 2—5.
8. *И. Яглом.* Необыкновенная алгебра. М., 1968.

IX класс

1. *С. Абрамов.* Поиск в упорядоченной совокупности и упорядочение // Квант. 1986. № 1.
2. *Л. Залманзон.* Беседы об автоматике и кибернетике., М., 1985.
3. Избранные вопросы математики. IX класс. М., 1979.
4. Искусство программирования // Квант. 1982. № 2, 3; 1983. № 2.
5. *Н. Криницкий.* Алгоритмы вокруг нас. М., 1984.
6. Полупроводниковые элементы вычислительной техники // Квант. 1985. № 9—12; 1986, № 1—6.
7. *С. Шапиро.* Решение логических и игровых задач (логико-психологические этюды). М., 1984.

X класс

1. *С. Абрамов.* Математические построения и программирование. М., 1978.
2. *Ф. Бауэр., Г. Гооз.* Информатика. М., 1976.
3. *А. Брудно, Л. Каплан.* Олимпиады по программированию для школьников. М., 1985.
4. *Н. Вирт.* Алгоритмы + структуры данных-программы. М., 1985.
5. *Н. Виленкин.* Метод последовательных приближений. М., 1968.
6. *Е. Габович.* Задача коммивояжера // Квант. 1978. № 6.
7. *Ю. Гильдерман.* Вооружившись интегралом... Новосибирск, 1980.
8. *У. Дал, Э. Дейкстра, К. Хоор.* Структурное программирование. М., 1975.
9. *Н. Демидович.* Вычисления, ошибки, контроль // Квант. 1973. № 2.
10. *В. Евстигнеев.* Графы и программы // Квант. 1981. № 3.
11. *Я. Зельдович.* Высшая математика для начинающих. М., 1968.
12. Избранные вопросы математики. X класс. Факультативный курс под ред. В. Фирсова. М., 1980.
13. *Э. Любимский и др.* Программирование. 1980.
14. *Ю. Нивергельт. Форрар Дж., Рейнгнолд Э.* Машинный подход к решению математических задач. М., 1977.
15. *М. Рейтман.* Транспортная задача // Квант. 1974. № 7.
16. *М. Рейтман.* Динамическое программирование // Квант. 1972. № 3.
17. *С. Пулькин.* Вычислительная математика. М., 1974.
18. *А. Строгова.* Метод наименьших квадратов // Квант. 1978. № 12.
19. *А. Тихонов, Д. Костомаров.* Рассказы о прикладной математике. М., 1979.
20. *Э. Тьмеладзе.* Нелинейное программирование // Квант. 1976. № 1.
21. *М. Цаленко.* Комбинаторные задачи информационного поиска // Квант. 1979. № 12.
22. *Ю. Шрейдер.* Что такое расстояние. М., 1969.
23. *Р. Шейнцвит.* Ветви и границы // Квант. 1972. № 7.
24. *В. Шкурба.* Задача трех станков. М., 1976.
25. *Ч. Уэзерелл.* Этюды для программистов. М., 1982.

К сведению читателей

2 и 3 сентября 1987 г. Минпросом СССР совместно с Гостелерадио СССР были проведены экспериментальные передачи компьютерных учебных программ по каналам центрального телевидения. В эфир по II общесоюзной программе были выданы тестовые регистрационные карточки на языках Фокал, Бейсик и программа шахматной игры в машинных кодах для ЭВМ БК-0010. Объем переданных данных около 32К байт. В контрольных точках Москвы прием прошел успешно. Повторение эксперимента 4 сентября не увенчалось успехом по причине неверно сформированной кассеты. Работа проведена группой специалистов в составе: А. Денисенко (Управление информатики Минпроса СССР), А. Зильберман (Гостелерадио), С. Яновский (Центр компьютерного обучения Ворошиловского района г. Москвы). Эксперименты предполагается продолжить.

С. БЕШЕНКОВ, Э. КУЗНЕЦОВ

О программе факультативного курса

«Основы компьютерного подхода к решению задач (34 часа)»

По решению коллегии Минпроса СССР педагогам-практикам разрешается разрабатывать программы факультативных курсов с последующим утверждением педсоветом школы.

Желая привлечь учителей к разработке собственных, мы приглашаем всех к обсуждению имеющихся в портфеле редакции программ. Предлагаем опробовать их на факультативных занятиях, а также создать свои программы и, если они покажутся вам удачными, прислать нам. Лучшие работы будут опубликованы.

Внедрение электронной вычислительной техники во все сферы человеческой деятельности поставило перед школой задачу обучить школьников основам информатики, сделать их компьютерно грамотными. Основную роль в этом призван играть новый общеобразовательный предмет «Основы информатики и вычислительной техники». Вместе с тем задача обеспечения компьютерной грамотности предполагает и более детальное изучение отдельных вопросов этого курса.

В настоящее время школа располагает определенным набором факультативов по основам программирования и вычислительной техники, созданных в 60—70-х годах. Однако они не в состоянии решить задачи углубленного изучения основ информатики. В связи с этим возникает проблема разработки новых факультативных курсов по этому учебному предмету.

Один из основных разделов курса — «Построение алгоритмов для решения задач». Рамки обязательного курса не позволили всесторонне раскрыть эту тему. Важность ее объясняется тем, что она содержит по существу общий методологический подход к решению задач на ЭВМ: от постановки задачи до анализа результатов исполнения программы. Школьный курс информатики будет усиливаться в аспекте теории и работы с готовым программным обеспечением. Сама же «технология» решения задачи на ЭВМ в определенной степени отходит на второй план. Тем не менее изучение этих вопросов представляется важным для всего курса информатики. Особенно это полезно для тех учащихся, которые в дальнейшем будут работать с вычислительной техникой. В частности тем, которые будут принимать участие в разработке учебного программного обеспечения.

С другой стороны, отсутствие задачника по обязательному курсу основ информатики и вычислительной техники может быть восполнено путем систематизации задач в данном факультативном курсе. Ниже приведены примерные содержание и структура такого курса. Они могут служить ориентиром для учителя при подготовке и проведении занятий. Отдельные фрагменты курса проходили экспериментальную проверку в школах № 57, 1140 Москвы. Отмечая положительные результаты, авторы вместе с тем считают, что рамки эксперимента должны быть расширены.

Структура курса

Курс открывается кратким введением, в котором дается представление об оформившейся в XX веке науке информатике, понятии информации, средствах ее обработки.

Первая часть — «Основы автоматической обработки информации» — знакомит с последовательностью компьютерного решения задач с последующим раскрытием каждого из этапов.

Значительно полнее, чем в действующем курсе, рассматриваются понятия модели, алгоритма, данных. В качестве языка программирования рассматривается Паскаль. Он является языком профессионального программирования и одновременно вполне доступен учащимся [4].

Во второй части — «Методы и алгоритмы» — излагаются наиболее распространенные алгоритмы, которые могут быть использованы при решении задач, связанных с обработкой информации. К ним относятся: алгоритмы поиска корня, численное интегрирование, решение простейших дифференциальных уравнений с помощью ломаных Эйлера, различные методы сортировки и т. д.

Эта часть является вспомогательной по отношению к первой части. Ее цель — сообщить учащимся ряд методов, типичных для решения прикладных задач.

Данный курс задуман как факультатив с обширными межпредметными связями. Этому способствует набор моделей и связанных с ними задач (см. приложение). Многие из примеров могут быть использованы при изучении обязательного курса основ информатики и вычислительной техники.

Краткие методические указания

Данный факультативный курс тесно примыкает к обязательному. Поэтому в целом методические подходы к изучению большинства тем могут быть заимствованы из методики преподавания основного курса. Вместе с тем изложение многих вопросов требует дополнительной подготовки (особенно в вопросах, связанных с понятием модели).

Первую главу курса целесообразно изложить учащимся полностью. Цель заключается в том, чтобы они при этом получили целостное представление о компьютерном подходе к решению задач. Содержание же второй главы может сообщаться учащимся по мере необходимости, при решении конкретной задачи. При отсутствии в школе кабинета вычислительной техники желательно организовать практикум на больших ЭВМ.

Предлагаемый факультативный курс целесообразно проводить параллельно обязательному курсу основ информатики и с опорой на него.

Это один из первых факультативов, связанных с действующим курсом основ информатики. Есть еще много вопросов и нерешенных проблем, и авторы с благодарностью воспримут все высказанные критические замечания.

Программа

Введение. Информатика — наука XX века. Понятие информации. Задачи обработки информации. Средства обработки информации (1 ч).

Основы автоматической обработки информации (24 ч).

Задачи, связанные с обработкой информации. Примеры. Этапы решения задач: постановка, построение модели, построение алгоритма, исполнение, анализ результатов (2 ч).

Моделирование — важнейший этап решения задачи. Примеры моделей. Решение задач (4 ч). Алгоритм решения задачи. Оценка сложности алгоритма. Использование ЭВМ. Понятие данных. Структура данных. Решение задач (8 ч).

Язык программирования — средство общения с ЭВМ. Элементы языка Паскаль (6 ч).

Исполнение программы на ЭВМ. Принципы фон Неймана (2 ч).

Анализ результатов. Уточнение модели. Решение задач (2 ч).

Методы и алгоритмы (9 ч).

Поиск корня (деление пополам, метод хорд и касательных, метод Ньютона). Численное интегрирование (формула прямоугольников, формула Симпсона). Решение дифференциальных уравнений (ломаные Эйлера). Методы сортировки. Оценки числа сравнений при сортировке, метод Монте-Карло.

Приложение. Примерный набор моделей и связанных с ними задач.

1. Модель маятниковых часов (Круговой маятник, циклоидальный маятник). Составление и решение дифференциальных уравнений [7].
2. Экологические модели. Составление и решение уравнений [7].
3. Химические реакции. Составление и решение дифференциальных уравнений [7].
4. Модели спроса и предложения. Составление и решение дифференциальных уравнений [7].
5. Транспортная задача. Поиск алгоритма наибольшей пропускной способности [3].
6. Статистические модели. Модель «вертушки» [3].

Литература

1. Основы информатики и вычислительной техники / под ред. А. П. Ершова и В. М. Монахова, ч. 1. М., 1985.
2. Изучение основ информатики и вычислительной техники / под ред. А. П. Ершова и В. М. Монахова, ч. 1. М., 1985.
3. *Нивергельт Ю., Фаррар Дж., Рейнгольд Э.* Машинный подход к решению математических задач. М., 1977.
4. *Дейкстра Э.* Дисциплина программирования. М., 1978.
5. *Вирт Н.* Систематическое программирование. М., 1977.
6. *Демидович Б. П., Марон И. А.* Основы вычислительной математики. М., 1966.
7. *Амелькин В. В.* Дифференциальные уравнения в приложениях. М., 1987.

ЧТО МОЖЕТ ЭВМ?

И волки сыты, и овцы целы

Современная бытовая радиоаппаратура нафарширована электроникой, зачастую весьма сложной. Например, на приобретающие все большую популярность лазерные компакт-диски музыка записывается цифровыми сигналами, и проигрыватели для них — настоящие цифроаналоговые вычислительные машины; но зато какое качество!

Уже несколько лет назад появились и экспериментальные кассетные магнитофоны, использующие цифровую запись и дающие прекрасный звук, однако в серию они не пошли. Причина — претензии фирм, выпускающих компакт-диски. Ведь владелец цифрового «кассетника» мог бы делать с компакт-диска неограниченное число копий без потери качества, а это чревато организацией подпольных студий звукозаписи и уменьшением доходов «граммофонных» фирм.

Но вот, наконец, японская компания «Айва» выпустила в продажу первый цифровой «кассетник». С частотами дискретизации записи 48 и 32 кГц он работает как магнитофон, а с частотой 44,1 кГц (стандартной для компакт-дисков) может только воспроизводить: при попытке записи включается режим «Пауза» и загорается индикатор «Копирование запрещено».

Вот сколько сложностей из-за хорошего звучания!

Быстро и модно

Выпуск нового изделия — проблема для легкой промышленности: изменение технологии, перестройка конвейера... А между тем мода диктует постоянные перемены в одежде. Как не отстать от нее и в то же время обойтись без издержек производства?

Вы уже догадались, что решение проблемы — в использовании компьютера.

На фабрике «Берлинская мода» (ГДР) используется несложная ПЭВМ, которая не только помогает создавать рисунок трикотажных изделий, цвет, раскрой, но и составляет программу для вязального автомата, а параллельно выполняет технико-экономические расчеты. Теперь запуск в производство новой модели занимает 8—10 дней, и за год ассортимент фабрики обновляется на 70—80 %.

КАБИНЕТ ВТ

В. НОВИЧКОВ, А. ПЫЛЬКИН

Рекомендации по оценке качества прикладных программ

Введение

Широкое использование вычислительной техники во всех сферах человеческой деятельности приводит к постоянному общению человека (пользователя) с машиной. Одним из средств этого общения является программа. От того, насколько она хороша, зависит успешность применения ЭВМ. Создание любой программы требует затрат времени, сил и средств. Поэтому возникает естественное желание использовать уже написанные и отлаженные программные средства при решении своих задач. Пользователь оценивает такие свойства, как компактность, быстродействие, понятность программы и др. Чаще всего при оценке качества рассматривают сразу несколько характеристик, которые могут быть противоречивыми — улучшение одной их них приводит к ухудшению другой. В этих условиях бывает трудно сформулировать однозначный критерий выбора «хорошей» программы.

Рассмотрим наиболее часто используемые характеристики качества программного обеспечения:

документированность;
эффективность;
надежность;
простота и удобство эксплуатации;
мобильность;
совместимость;
стоимость (определяется нормативными документами).

Существуют три типа программ, и для каждого оценка качества должна быть различной.

К первому типу относятся операционные системы, компиляторы, библиотеки стандартных программ. Для них главной характеристикой является эффективность.

Второй тип составляют производственные программы, используемые длительное время. Хотя эффективность в данном случае и существенна, но обычно больше внимания уделяется их эксплуатационным характеристикам. Как правило, первый и второй типы программы создаются профессиональными программистами.

Третий тип программ — программы, разрабатываемые непрофессионалами. Время для этих людей важнее всего. Здесь эффективность имеет значение только в том смысле, что программа должна уместиться в заданном объеме памяти и выполняться за приемлемое время.

Документированность

Знакомство с программными средствами (программами, пакетами и комплексами программ, системами и т. д.) начинается с чтения программной документации. Для уяснения назначения программы, порядка ее использования и возможностей необходимо, чтобы соответствующая информация была достаточно полно и понятно изложена в документах. Документы не должны содержать повторения одних и тех же сведений в разных разделах, ссылок на труднодоступные источники, без ознакомления с которыми трудно разобраться в программе. Текст программы должен быть ясным и понятным, с тщательно подобранными идентификаторами, лаконичными и информативными комментариями. Структура программы должна достаточно легко усматриваться из ее текста и т. д. Документация должна быть полной, т. е. содержать весь перечень установленных ЕСПД и надлежащим образом оформленных документов.

Документирование должно выполняться

одновременно с разработкой программы, начиная с этапа постановки задачи (хотя документирование и составляет самостоятельный этап решения задачи на ЭВМ). Даже если программа предназначена для индивидуального использования, следует составить инструкцию и описание, чтобы не забыть деталей, касающихся ее работы. Если же программа предназначена для непрограммистов, необходимы хорошо и подробно составленные инструкции.

Плохо документированная программа является «вещью в себе», разобраться в ней бывает чрезвычайно сложно. Нет ничего удивительного в том, что такие программы обычно выбрасывают или переписывают заново.

Формальные требования к документации программ описаны в ЕСПД. В общем случае документация состоит из следующих материалов:

1. Описание задачи.
2. Область применимости программы.
3. Описание метода.
4. Режим работы программы (точки входа в процедуру, сообщения программы, ответы оператора).
5. Исходные данные, выдаваемые результаты, побочный эффект.
6. Правила подготовки исходных данных на внешних носителях и вид выдаваемой информации.
7. Описание структуры данных программы.
8. Схема алгоритма (блок-схема, структурограммы, псевдокод и пр.).
9. Текст программы (в виде распечатки с ЭВМ).
10. Тексты для отладки.
11. Возможные модификации программы.

Эффективность

Понятие эффективности программы весьма трудно определить как некоторую абсолютную величину, поскольку она в значительной мере зависит от назначения программы, конфигурации аппаратных средств, общесистемных программ и др. В первом приближении за оценку эффективности принимают характеристику программы, которая прямо пропорциональна быстродействию и обратно пропорциональна объему используемых ресурсов. Быстродействие программы при документировании заменяется заданием времени ее выполнения. В качестве ресурсов программы могут выступать оперативная память, внешняя память, количество типов и единиц внешних устройств, частота обращений к внешним устройствам, время обслуживания оператором и т. д.

Лишь в особых случаях эффективность выступает в качестве главного требования: программа либо не помещается в памяти, либо слишком долго выполняется. В этом случае отдают предпочтение эффективности в ущерб удобочитаемости.

Надежность

Надежность прежде всего подразумевает отсутствие ошибок в программе, но поскольку ошибки неизбежны, то программа должна быть организована таким образом, чтобы возможные ошибки не вели к фатальным последствиям и могли быть быстро исправлены.

Исследования многих авторов дают примерно одинаковую частоту ошибок в сложных программах. Она составляет в среднем 0,01—0,02, т. е. из 100 строк программы в среднем одна-две содержат ошибку. В связи с этим любая прикладная программа должна быть тестирована или просчитана на контрольных примерах, а также проверена на правильность при вырожденных случаях.

Простота и удобство эксплуатации

Для каждой программы должны быть четко определены условия и области применения с ясно оговоренными ограничениями, облегчающими и ускоряющими использование, но исключающими неправильное применение. Должны быть четко определены входные параметры, форматы ввода данных, описание выходов должно быть ясным и легко интерпретируемым, позволяющим гибко определять формат и содержание результирующей информации.

Поскольку при создании программы трудно гарантировать, что она является совершенной, то в ней должны быть предусмотрены возможности для дальнейшего улучшения и разумной модификации без больших переделок, с тем чтобы эксплуатация программы осуществлялась с минимальными затратами. Программу нельзя считать удобной, если в качестве входных данных определены значения общепринятых констант (например, число Архимеда π, основание натуральных логарифмов e, гравитационная постоянная γ, ускорение свободного падения g и т. д.), не производится эхо-печать исходных данных, на печать выводятся только значения переменных без вспомогательного текста и пр.

Мобильность и совместимость

Мобильность характеризует приспособленность программного обеспечения к переносу с одной машины на другую, а также к смене операционной системы. При этом, естественно, нельзя отделять перенос программы от переноса данных. Быстрое развитие вычислительной техники, смена поколений ЭВМ, высокая стоимость математического и программного обеспечения выдвигают эту характеристику в число важнейших. Практика написания программ на машинных языках и близких к ним фактически лишает их свойств мобильности. Нельзя назвать программу мобильной, если она написана на одном из «экзотических» языков или явно устаревшем.

Отдельные характеристики качества являются взаимозависимыми, т. е. улучшение одной характеристики приводит, как правило, к ухудшению другой. В то же время количественная оценка отдельных характеристик весьма затруднена (или совсем невозможна). В этих условиях единственно возможным способом оценки качества программ является экспертная оценка качества, осуществляемая по результатам опроса мнений специалистов. Несколько экспертов дают оценки отдельных характеристик (например, по 10-балльной системе). Затем подсчитывается среднее значение характеристики и путем суммирования находится обобщенная характеристика качества программы. Программу следует считать «плохой», если хотя бы одно из средних значений меньше 5. Если все средние характеристики больше 5, то их сумма (или еще раз усредненное по числу характеристик значение) может служить обобщенной характеристикой качества программы.

Правильный выбор языка программирования, наиболее подходящего для решения задачи, в значительной степени определяет успех и качество проделанной работы. Язык программирования должен облегчать установление соглашений и ограничений, позволяющих объединить в единое целое части большой программы, разрабатываемые отдельными лицами или группами, обеспечивать простоту и наглядность программ, написанных и отлаженных одним лицом, для других пользователей.

На нижней ступени в иерархии языков стоят языки микропрограммирования. Каждая микрокоманда в микропрограмме описывает элементарный акт переработки информации вычислительной машиной. Пользователи ЭВМ редко пишут микропрограммы. Микропрограммирование, как средство, предоставляемое пользователю, целесообразно в ряде специальных случаев, например, для ЭВМ, входящих составной частью в систему управления.

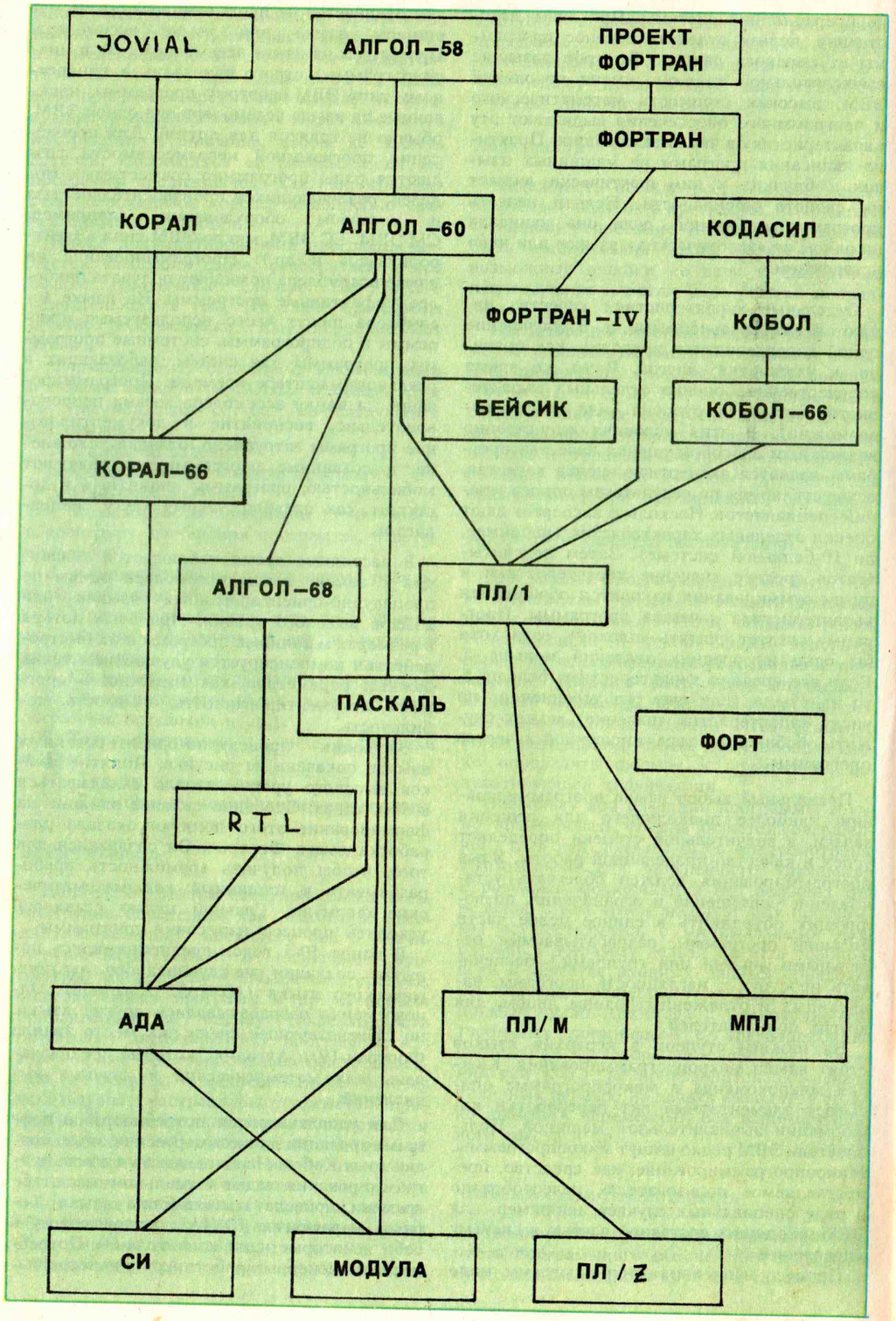
Процесс написания микропрограмм мало чем отличается от программирования на машинном языке или языке ассемблера. Программы на языке ассемблера (как и микропрограммы) строго привязаны к конкретному типу ЭВМ, поэтому программы, написанные на языке ассемблера для одной ЭВМ, обычно не годятся для другой. Для преодоления программной несовместимости создаются ряды программно совместимых моделей, различающихся производительностью и составом оборудования (например, СМ ЭВМ, ЕС ЭВМ, микро-ЭВМ типа «Электроника-60» и др.). Программирование на языке ассемблера позволяет получать наиболее эффективные программы. На языке ассемблера пишут часто используемые программы и подпрограммы, системные программы, программы для систем, работающих в реальном масштабе времени. Программирование на языке ассемблера весьма непроизводительно, восприятие и документирование программ затруднено, отладка усложнена, а созданные программы не обладают мобильностью, программы «понятны» и доступны, как правило, узкому кругу специалистов.

В настоящее время наблюдается тенденция к замене языка ассемблера одним из процедурно-ориентированных языков или языков высокого уровня. При этом потеря в размерах машинных программ и их быстродействия компенсируется улучшением таких важных параметров, как надежность программ, документированность, стоимость, мобильность.

Эволюция процедурно-ориентированных языков показана на рисунке. Понятие языков высокого уровня начало складываться в 50-х годах. Особенно сильное влияние на формирование этого понятия оказала разработка языка Фортран. Он создавался для того, чтобы получить возможность преобразовывать в машинный код математические формулы. Главной целью ставилось ускорить процесс написания программы.

В конце 50-х годов предпринимались попытки создания международного алгоритмического языка (JOVIAL, Алгол-58). Одновременно разрабатывались другие языки, но главенствующее среди них место заняли Фортран-IV и Алгол-60, которые предназначены для математических и научных вычислений.

Для удовлетворения потребностей в программировании для коммерческих задач создан язык Кобол. Необходимость в языке программирования задач в реальном масштабе времени привела к появлению языка Коралл — развитию JOVIAL, включающему в себя некоторые идеи из Алгола-60. Потребность инженерных работников в интерактив-



ном режиме работы на ЭВМ породила основанный на Фортране-IV язык Бейсик.

Язык ПЛ/1 является попыткой совместить все лучшее, имеющееся во всех основных языках программирования (Фортран, Алгол, Кобол). Главная цель: иметь в языке средства, необходимые всем категориям программистов, с тем чтобы отдельному программисту достаточно было ознакомиться лишь с определенным подмножеством языка. Одновременно с универсальным языком ПЛ/1 развивались другие языки (Алгол-68, язык реального времени RTL/2).

Дальнейшим толчком к появлению новых языков послужили появившиеся к этому времени микропроцессоры, микро- и мини-ЭВМ. Так, были созданы языки ПЛ/М, ПЛ/Z, МПЛ. Ряд языков получил «второе дыхание» (самый яркий тому пример — Паскаль). Впервые появившись в 1969 г., Паскаль применялся для обучения программированию. Благодаря распространению на микро-ЭВМ язык Паскаль в настоящее время стал одним из самых известных и популярных языков программирования. Еще одним «воспрянувшим» языком стал Бейсик. Первоначально он создавался для того, чтобы сделать работу за терминалом ЭВМ похожей на работу с обычным калькулятором. Он наиболее прост в изучении и использовании. Пристальное внимание на этот язык обратили в период широкого внедрения персональных микро-ЭВМ. В настоящее время модернизированная версия Бейсика, вероятно, наиболее широко используемый язык на микропроцессорах и микро-ЭВМ.

В конце 70-х и начале 80-х годов появились новые (и более совершенные) языки программирования — Ада, Си, Модула и др. Возможности этих языков программирования достаточно широки. Например, язык Си использован при написании операционной системы (ИНМОС). Новейшие языки программирования позволяют создавать более качественное программное обеспечение, но широкий круг пользователей «запаздывает» с их освоением.

Программы, которые создаются в процессе обучения, должны отвечать новейшим достижениям в области программирования (быть структурированными, написанными на перспективных языках программирования, отличаться хорошим стилем, иметь модульную структуру, использовать существующие технологии программирования) и при этом быть простыми и понятными. Ничто так отрицательно не сказывается на обучении, как плохо составленная и оформленная программа.

А. МИРСКАЯ, Т. СЕРГЕЕВА

Обучающие программы оценивает практика

Сейчас большинство компьютерных обучающих программ (КОП)* создается на основе индивидуального опыта программистов, их, часто интуитивного, представления о том, какой должна быть КОП. Однако для успешного создания этих программ необходимо сотрудничество педагогов, психологов и программистов; нужны четко сформулированные требования к КОП. В работах ряда исследователей мы находим такие требования, представленные на различных уровнях: методологическом [5, 11, 16], дидактическом [9, 14], психологическом [1, 12, 17], техническом [7], физиолого-гигиеническом [10]. Наша задача состояла в том, чтобы при экспериментальном использовании КОП в учебном процессе уточнить и дополнить эти требования. (Экспериментальная работа проводилась в школе № 1140 с учащимися VI, VII, IX классов, посещающими факультативные занятия по информатике). Полученные нами результаты позволили составить ряд рекомендаций, которые, как мы надеемся, будут способствовать созданию более эффективных обучающих программ.

Обратимся к некоторым требованиям, предъявляемым к структуре КОП. В программах обучающего типа мы выделяем три основных структурных компонента: ориентировочный, исполнительный и контрольно-корректировочный.

Ориентировочная часть содержит различные справочные сведения, необходимые для успешной работы с программой. Сюда входят титульный лист, формулировка цели выполняемого задания, описание задания,

* Под компьютерными обучающими программами мы понимаем такое средство обучения, которое является носителем определенного предметного содержания, либо представляет объекты изучения и, организуя деятельность как преподавателя, так и учащихся, может быть реализовано в учебном процессе только с помощью ЭВМ.

правила работы с программой и другой справочный материал.

Информация *титульного листа* необходима как учителю, так и учащимся. Как правило, это название программы, учебного предмета, указание класса, а также при какой форме занятий данную КОП наиболее целесообразно использовать (урок, факультативное занятие, домашняя подготовка и т. д.). Желательно стандартизировать представление этой информации, обеспечив возможность быстрого считывания титульных листов всех КОП, имеющихся на дискете.

Формулировка *цели задания* является чрезвычайно важным элементом ориентировочной части. Она должна быть четкой, конкретной и лаконичной. В конце работы с программой необходимо указать, достигнута ли учащимся поставленная цель.

Описание задания — это краткое изложение тех действий, которые должны быть осуществлены учеником при работе с программой. Например, в КОП репетиторского типа по химии «Классификация неорганических соединений» задание заключается в следующем: «Из приведенного списка формул веществ выберите формулы оксидов, оснований, кислот и солей и в соответствующем столбце напротив формулы поставьте знак «+». Если программа имеет несколько уровней сложности, то при общей цели задания могут различаться по сложности.

Правила работы с программой (пояснение способов ответа на вопросы и перехода к другим частям программы, значение появляющихся на экране символов и т. д.). Произвольное кодирование правил недопустимо. Наиболее естественный путь — это путь мнемических средств, например использование клавиш со стрелками (↑, ↓, →←) для указания направления перемещения курсора или некоторого объекта на дисплее, использование клавиш с буквой, соответствующей первой букве слова (П — правило, К — комментарий, С — справка, Д — да, Н — нет). Не стоит использовать кодирование, привычное только для программистов: 1 (да) и 0 (нет), а также заимствования из английского языка.

Наблюдая за работой учащихся, мы убедились в необходимости обеспечить вызов правил пользования программой в любой момент, при постановке любого вопроса. Возможно и такое построение программы, когда информация о правилах работы входит в формулировку вопроса. Желательно также, чтобы учащиеся могли возвратиться к предыдущему кадру или прекратить работу программы нажатием на определенные клавиши. В некоторых КОП, нацеленных на отработку типового умения, целесообразно дать образец выполнения задания.

Справочная информация (правила, формулы, табличные данные и т. д.) — это изложенный в конспективной форме материал, необходимый школьнику при работе с программой. КОП должна обеспечить возможность обращения к этой информации в любой момент по желанию учащегося.

При работе с программой ученик должен знать, что делать в данный момент времени, а для этого необходима поясняющая информация на экране в виде указаний очередных действий. Например, информация о том, что для перехода к следующему этапу работы надо нажать клавишу «пробел», обычно дается в виде справочной информации в начальных кадрах. Многие учащиеся в процессе работы забывали это указание. Следовательно, такую информацию целесообразно давать в каждом кадре.

Нами отмечено (подобный факт подтверждается и другими авторами [4]), что начинающий пользователь нажимает на клавишу до тех пор, пока на экране не произойдет какое-либо изменение. Поэтому полученная от ученика информация должна сразу же приводить к изменениям на экране дисплея.

Исполнительная часть призвана организовывать исполнительную деятельность учащихся: решение задач, построение графиков, проведение опытов, моделирование. Следует помнить, что в программах моделирующего типа необходимо представлять *не чувственный объект, а его модель* (т. е. не увлекаться натурализмом). Компьютерная модель должна способствовать реализации дидактических целей данной программы; ее следует предъявлять в форме, позволяющей наиболее полно раскрыть существенные связи и отношения, которые в программе должны быть адекватно зафиксированы цветом, звуком, миганием символов и т. д.

В современных исследованиях отмечено [15], что эффективность обучения повышается, когда учащиеся сами строят модели, а не получают их в готовом виде. В процессе построения модели и на основе действий с нею учащиеся самостоятельно усваивают теоретические знания. Работа с моделью может вестись в двух направлениях: достраивание модели или ее видоизменение и переконструирование. Работа с КОП моделирующего типа только тогда эффективна, когда деятельность моделирования заранее сформирована (т. е. имеется представление о модели, моделировании, цели, объекте и т. д.).

Контрольно-корректировочная часть позволяет осуществлять контроль за деятельностью ученика и ее коррекцию.

При организации диалога с учеником используют два способа ввода ответов: 1) метод выборочных ответов (выбор правильного ответа из нескольких); 2) метод конструируемых ответов (произвольный ответ ученика). Первый из них имеет существенные недостатки, из которых основной — попытки учащихся угадать правильный ответ. Однако в некоторых случаях применение этого метода вполне оправданно. В частности, когда количество ответов ограничено самой задачей и все потенциально возможные ответы известны учащимся. Например, в программе по химии на вопрос: «К какому классу относится данное соединение?» — перечисляются все четыре класса веществ: кислоты, основания, оксиды, соли. Альтернатива вводится и для того, чтобы учащийся мог лучше ориентироваться в возможностях КОП. Например, перечисляются виды помощи, которая может быть оказана ученику, или этапы работы: демонстрационный режим, контроль и т. д.

Мы рекомендуем при использовании данного метода предъявлять для выбора только возможные, правильные ответы, избегая запоминания неверных.

При введении конструируемых ответов желательно предусмотреть максимальное количество вариантов ответа. Если в ответе встречаются грамматические ошибки, его следует обрабатывать как правильный, но учащемуся необходимо сообщить о грамматических недочетах. Если учащийся ввел неправильный ответ, то ему надо объяснить, в чем состоит его ошибка, и/или выдать необходимую справочную информацию.

Следует иметь в виду возможность ошибок, связанную со случайным нажатием на клавишу. При использовании метода выборочных ответов, о неправильном ответе целесообразно судить по второй попытке. Первый ответ можно рассматривать как «опечатку». При использовании метода конструируемых ответов необходимо предусмотреть возможность редактирования ответа.

Чтобы избежать распространения среди учащихся информации о номерах правильных ответов, при разработке контролирующих программ и программ-тренажеров желательно обеспечить случайный вывод вопросов (если это не противоречит логике программы).

На эффективность работы ученика с КОП в значительной степени влияет характер реплик и комментариев, которые должны отражать специфику деятельности. Подобная подача позволит фиксировать внимание на усваиваемом материале. От заполняющих паузы реплик требуется смысловой комментарий процессов, происходящих в ЭВМ. Поэтому вместо иногда используемых в таких случаях реплик «Минуточку», «Наберитесь терпения» и т. д. лучше давать пояснения: «Ввод программы», «Ввод данных с диска», «Обработка ответа». Иногда репликам и комментариям пытаются придать шутливый характер. По нашим наблюдениям, это отвлекает внимание учащихся, а при многократной работе с программой даже раздражает и утомляет.

В работах некоторых авторов [1, 6, 8] указывается на феномен антропоморфизма по отношению к компьютеру. Мы также с этим сталкивались, но только у взрослых пользователей. При работе со школьниками подобного явления не наблюдалось, но тем не менее при организации диалога с компьютером следует стремиться к тому, чтобы ученик понимал ограниченность возможностей машины. Диалог ученика с машиной не должен имитировать диалог ученика с учителем. В лексику программы не стоит включать понятия, характеризующие психологическую сферу деятельности человека (местоимение «я», глаголы «понимать», «думать» и т. п.).

После выполнения задания учащимся, ему необходимо представить информацию о допущенных ошибках, их количестве, возможной причине, времени выполнения задания (если последнее является оценочным критерием) и т. д. Наш опыт позволяет утверждать, что перечень ошибок должен носить не столько количественный, сколько качественный характер. Целесообразно не просто перечислить допущенные ошибки, а указать, какой материал учащийся усвоил недостаточно хорошо, что ему следует повторить, на что обратить особое внимание. В КОП должна быть также представлена итоговая информация для учителя (как правило, это статистически обработанные данные о работе учащегося с программой и результатах этой работы).

Вопросы методики работы с программой. Сформированность мотивационной сферы учащегося и поддержание соответствующей мотивации является необходимой предпосылкой эффективности обучения. Решающее значение имеет то, *что* именно побуждает школьников к учебе, *каковы мотивы* учебной деятельности. Наши наблюдения подтверждают точку зрения других исследователей [13] о том, что задания должны быть социально или личностно значимыми. Подобный подход способствует скорейшему формированию познавательных мотивов учебной деятельности.

Интересен и такой факт: работая со школьниками, изучающими ОИВТ первый и второй год, мы наблюдали, что отношение к эстетике внешнего оформления у новичков и у учащихся второго года обучения различно. Для новичков определяющими являются внешние мотивы, различного рода эстетические «поиски» программиста; игровые ситуации являются достаточно сильным стимулом в работе. Большинство учащихся второго года обучения руководствуется собственно *познавательным интересом,* внешние мотивы уже не являются определяющими. Если четко сформулирована учебная задача, имеющая социальное или личностно значимое содержание, то учащиеся предпочитают эту задачу игре.

Замечено, что на завершающем этапе работы с КОП у обучаемых появляется потребность в конкретном результате своей деятельности: программе, записанной на диск, или распечатке, выполненной на печатающем устройстве. Обмен распечатками программ, их сравнение, обсуждение принимают характер сотрудничества.

При разработке КОП необходимо учитывать психофизиологические особенности учащихся, в частности индивидуальный стиль деятельности. Одни быстро решают поставленную перед ними задачу, но при этом часто ошибаются. Другие долго думают над задачей, тщательно взвешивают ответ, проверяют решение, допуская мало ошибок. Наши наблюдения подтверждают выводы советских исследователей [2, 3] о том, что КОП должна быть «гибкой», рассчитанной на учащихся с различным когнитивным стилем, типом восприятия, темпераментом, мышлением и др. Так, одним учащимся можно предложить выполнить определенное число упражнений в строго заданное время; других нежелательно ограничивать во времени.

Советы по представлению информации на экране дисплея. Известно, насколько велики возможности компьютера при представлении информации — текст, графика, движущиеся объекты, цвет, звук. Одна из основных задач при составлении КОП — максимальное использование этих возможностей. Мы наблюдали за тем, как учащиеся воспринимают различные формы представления информации на экране дисплея. Анализ результатов наблюдений позволяет высказать некоторые пожелания. Для привлечения и фиксации внимания целесообразно использовать мигающие символы (число миганий — 3—4 раза, период мигания — 0,5—1 с), крупный шрифт, подчеркивание отдельных слов, надписи на контрасном по отношению к цвету экрана фоне, звуковой сигнал. При этом нельзя забывать, что мигающие символы обладают и сильным отвлекающим эффектом. Использовать их желательно только в тех случаях, когда необходимо прервать работу учащегося (например, сообщить об ошибке). Каждому учащемуся требуется разное время для восприятия информации с экрана, поэтому в КОП переход к следующему кадру должен осуществляться по *команде ученика.*

Небольшое количество текста легче воспринимается на экране дисплея. Если же необходимо одновременно вывести на экран большой объем информации, то текст желательно представлять на разных цветных полосах или шрифтом разных цветов. В результате наблюдений за работой школьников мы получили данные, коррелирующие с экспериментальными данными НИИ ОиПП АПН СССР, полученными для взрослых пользователей ЭВМ: на экране должно одновременно присутствовать не более четырех цветов, имеющих определенные значения (красный — прерывание, экстренная информация, опасность; зеленый — разрешающий и т. д.).

При создании и использовании КОП нами были определены наиболее удачные сочетания цвета букв и фона: темно-зеленый фон — белые буквы; бледно-голубой — черные; синий или черный — белый текст в темно-зеленых рамочках; черный — зеленый, красный, синий текст; командная строка: на фиолетовой полосе — белые буквы; серый фон, черные буквы, красный график; светло-голубая рамка, черный фон, желтые буквы, зеленый график. Трудно читать текст, написанный зелеными буквами на черном фоне или на зеленом — черными буквами.

На наш взгляд, при групповых формах организации учебной деятельности определенное педагогическое воздействие, стимулирующее соревновательный элемент учения, могут иметь звуковые сигналы, указывающие на ошибку.

Еще раз подчеркнем, что об эффективности КОП можно судить только на основе ее *апробации в учебном процессе.* Поэтому решающее значение в проведенном исследовании придавалось наблюдениям за работой учащихся.

Литература

1. *Бабаева Ю. Д., Войскунский А. Е., Кобелев В. В., Тихомиров О. К.* Диалог с ЭВМ: психологические аспекты // Вопросы психологии. 1983. № 2. С. 25—34.

2. *Брусенцова Т. Н.* Исследование когнитивных стилей учащихся в автоматизированной системе обучения // Вопросы психологии. 1984. № 4. С. 71—76.

3. *Брусенцова Т. Н.* Опыт исследования когнитивных стилей в компьютерной системе обучения «Наставник» // Психолого-педагогические и психофизиологические проблемы компьютерного обучения. М., 1986. С. 62—86.

4. *Виллемс А., Пейал Я.* Советы по созданию дружественного программного обеспечения // Информатика и образование. 1986. № 1. С. 76—77.

5. *Гершунский Б. С.* Прогностический подход к компьютеризации // Советская педагогика. 1986. № 7.

6. *Дрябин В., Кочетков Г.* Игры и дела персонального компьютера // Наука и жизнь. 1986. № 9. С. 64—70.

7. Кабинет вычислительной техники всех типов средних учебных заведений (на базе персональных микро-ЭВМ): Методические рекомендации // Информатика и образование. 1986. № 3. С. 5—17.

8. *Кочетков Г. Б.* Американцы и персональный компьютер // США: экономика, политика, идеология. 1986. № 4. С. 46.

9. *Кузнецов А., Сергеева Т.* Обучающие программы и дидактика // Информатика и образование. 1986. № 2. С. 87—90.

10. *Леонова Л., Савватеева С.* Занятия в КВТ: влияние на организм // Информатика и образование. 1986. № 3. С. 82.

11. *Машбиц Е. И.* Компьютеризация обучения: проблемы и перспективы // М., 1986.

12. *Машбиц Е. И.* Психолого-педагогические аспекты компьютеризации // Вестник высшей школы. 1986. № 4.

13. *Панюшкин В. П.* Два подхода к обучению школьников программированию на ЭВМ // Психолого-педагогические и психофизиологические проблемы компьютерного обучения. М., 1984. С. 41—52.

14. *Роберт И.* Какой должна быть обучающая программа? // Информатика и образование. 1986. № 2. С. 90—95.

15. *Салмина Н. Г.* Виды и функции материализации в обучении. М., 1981.

16. *Талызина Н. Ф.* Внедрению компьютеров в учебный процесс — научную основу // Советская педагогика. 1985. № 12.

17. *Уайт М. Э.* Микроэлектроника: революция в обучении // Перспективы. 1985. № 4. С. 5—14.

П. МЕНЯЙЛО, М. ЩЕКОЧИХИН
г. Новосибирск

Полиэкранный графический редактор

Реформа среднего и среднего специального образования ставит перед советской школой задачу обеспечения компьютерной грамотности молодежи. Одним из направлений этой работы является постоянное совершенствование методического обеспечения процессов компьютеризации в школе, а следовательно, обеспечение высокого качества разрабатываемого компьютерного дидактического материала (КДМ). Необходимым условием повышения качества КДМ является более полное использование графических возможностей ПЭВМ. В процессе разработки КДМ, основанных на графических возможностях персональных компьютеров, в последнее время все шире используются графические инструментальные средства.

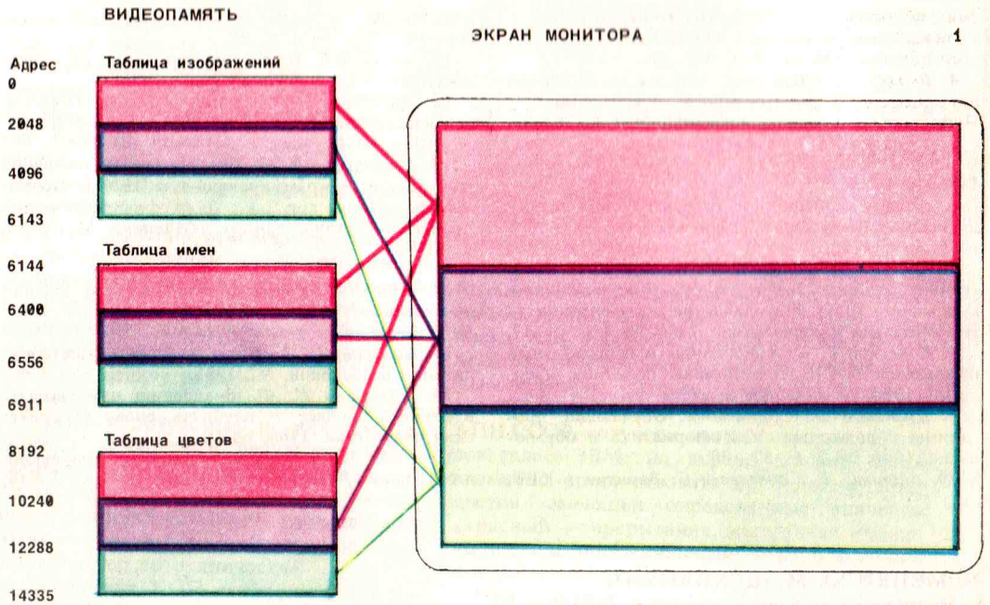
В настоящее время наиболее распространенные подходы, связанные с разработкой графических редакторов, основываются преимущественно на моноэкранном принципе. Данный принцип не позволяет в ряде случаев делать графическую информацию достаточно наглядной и детальной. В то же время графические средства современных ПЭВМ дают возможность реализовать полиэкранный принцип. Последний заключается в использовании графического поля, превышающего по площади размеры экрана. Этим достигается достаточно детальное отображение графической информации.

В НИИ ИВТ АПН СССР в 1986 г. авторами статьи разработан полиэкранный графический редактор GRED для ПЭВМ «Ямаха».

Целью настоящей статьи является описание аппаратных графических средств ПЭВМ «Ямаха», методики создания полиэкранного графического редактора, а также оценки возможностей и перспектив использования графической системы в учебном процессе.

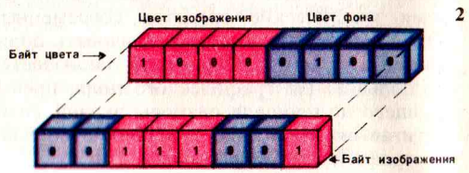
Графические возможности ПЭВМ «Ямаха» поддерживаются аппаратным средством — видеопроцессором. Видеопроцессор имеет собственную оперативную память (видеопамять), которая доступна центральному процессору через порты ввода-вывода. Видеопроцессор обеспечивает отображение информации, хранящейся в видеопамяти, на экран графического дисплея. Для управления видеопроцессором служат несколько групп регистров: а) регистры установки режимов отображения информации (текстовые режимы — 0 и 1, графические режимы — 2 и 3); б) регистры управления подвижными графическими объектами (спрайтами); в) регистры управления цветом.

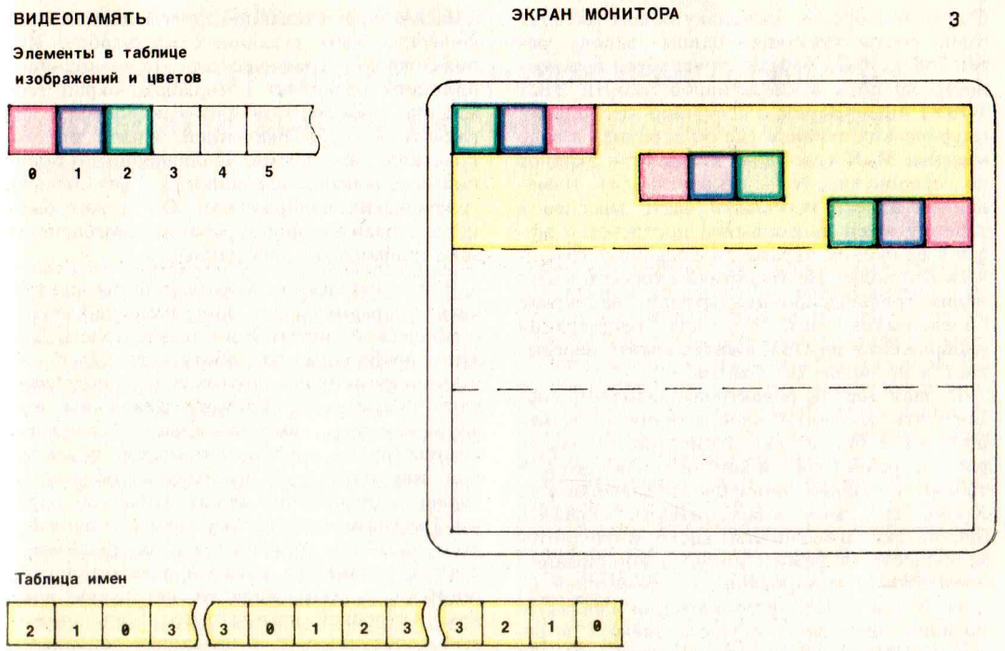
При разработке редактора использовался графический режим-2, обеспечивающий оптимальную разрешающую способность экрана. Видеопамять в этом режиме имеет табличную организацию и представлена таблицами: изображений, цветов, имен; каждая



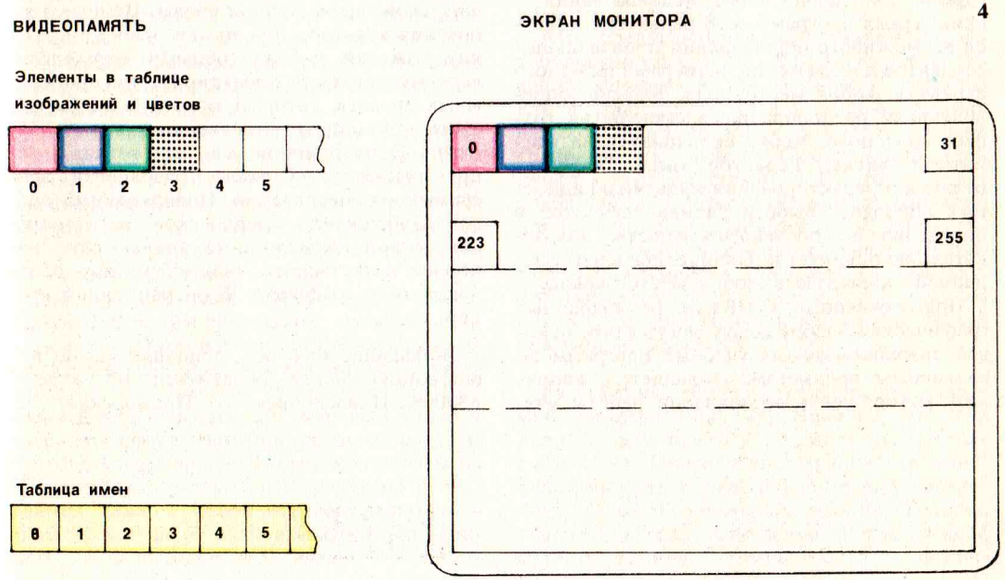
разделена на 3 равные части. В соответствии со структурой таблиц экранное поле также представлено тремя горизонтальными окнами. Изображение в каждом окне кодируется участками таблиц изображений, цветов и имен, относящимися к каждому из окон. Расположение таблиц в видеопамяти представлено на рис. 1. Каждой точке, отображаемой на экране, соответствует 1 бит в таблице изображений. Если бит установлен в положение «0», то соответствующая ему точка экрана является точкой фона, если в положение «1» — то точкой изображения. Так как видеопамять имеет байтовую организацию, то информация, находящаяся в таблице изображений, отображается на экране побайтно — линиями по 8 точек. Каждому байту таблицы изображений соответствует 1 байт в таблице цветов, причем «младшие» 4 бита кодируют цвет фона, а «старшие» — цвет изображения (рис. 2). Этим объясняется наличие одновременно на экране не более 16 цветов. Таким образом, на каждые 8 соседних точек изображения приходится два цвета: основной и фоновый (см. рис. 1). Попытка установить для этих восьми точек третий цвет приведет к переопределению уже имеющихся цветов. Это явление — перетекание цветов — возникает при неудачном расположении графической информации на экране.

Видеопроцессор отображает информацию на экране так, что восьми соседним байтам таблиц изображений и цветов соответствуют на экране монитора 8 линий, расположенных одна под другой. Эти линии образуют участок изображения из 64 точек размером 8*8. Этот участок изображения (элемент) имеет свой номер (имя). Номер элемента определяется расположением байтов, кодирующих этот элемент, в соответствующей трети таблицы изображений и цветов. Так, элемент, кодируемый 0—7 байтами, имеет номер 0; элемент, кодируемый 8—15 байтами, имеет номер 1 и т. д. Номера (имена) элементов заносятся в таблицу имен. После включения ЭВМ или при смене режима отображения информации происходит инициализация видеопамяти, во время которой в каждую треть таблицы имен последовательно заносятся значения от 0 до 255, что обусловливает расположение элементов на экране соответственно их естественному расположению в таблице изображений и цветов (рис. 3). При необходимости через порты ввода-вывода можно занести в любое место таблицы имен новое значение. Видеопроцессор при формировании изображения поместит в указанное место экрана тот элемент, номер которого соответствует значению, установленному в





таблице имен. При этом один и тот же элемент может отображаться на экране в нескольких местах (в пределах горизонтального окна экрана, для которого он определен). Обладая некоторым набором элементов, можно синтезировать различные графические изображения (рис. 4). Этот принцип положен в основу полиэкранного графического редактора. Так как расположение элементов графического изображения на экране определяется содержанием таблицы имен, смена содержимого таблицы имен позволяет быстро сменять изображение на экране, перемещать объекты по экрану, видоизменять их форму и т. д. Этим, в частности, достигается эффект мультипликации.



С другой стороны, поскольку объем таблицы имен, соответствующий одному экрану, равен 768 (256*3) байтам, появляется возможность хранить в оперативной памяти ЭВМ (ОЗУ) информацию о состоянии нескольких графических экранов (до 60 экранов) в виде массива M*N (где M — количество экранов по горизонтали, N — по вертикали). Изменением адреса пересылки части массива в таблицу имен видеопамяти достигается эффект движения экрана по большому графическому полю. Необходимая скорость изменения графической информации на экране определяется тем, что для регенерации изображения из ОЗУ в видеопамять пересылается не более 768 байтов.

Первая версия редактора позволяет формировать элементы изображения и размещать их в библиотеке элементов. Во время работы редактора библиотека находится в таблицах изображений и цветов видеопамяти. Кроме того, имеется возможность сохранять библиотеку на магнитном диске. В библиотеке существует режим прямого копирования элементов и копирования с симметрией по осям X или Y. При формировании и редактировании элементов, осуществляемых в режиме ЛИНЗА, элемент увеличивается в 64 раза. Подготовленные элементы пользователь может помещать в любом месте графического поля, синтезируя необходимое ему изображение. При этом в ОЗУ сохраняется лишь расположение элементов в графическом поле. Кроме того, пользователь может подготовить сегменты изображения (например, мультипликационные кадры) в произвольных участках поля и установить их относительные адреса для дальнейшего использования в прикладных программах. В редакторе имеется возможность перемещения произвольных сегментов изображения размерами до одного экрана в любое место поля. Массив, определяющий расположение элементов на графическом поле, также сохраняется на магнитном диске. Редактор имеет средства отладки и редактирования мультипликационных эпизодов. Выбрав размер сегментов и определив их последовательность, пользователь экспонирует выбранные сегменты с заданной скоростью в любом месте экрана.

Подготовленные в GREDe полиэкранные графические изображения могут стать основой разрабатываемых учебных программ по различным предметам. Появляется, например, возможность организации работы учащихся с использованием фрагментов географических карт различных масштабов. Использование графического полиэкранного принципа позволяет превращать экран терминала в «телескоп» или «микроскоп», заглянуть в глубь Вселенной, живой клетки, кристалла или атома. Полиэкранный принцип обеспечивает наглядность и детальность графических изображений. Он может быть использован для формирования многоаспектных графических баз данных.

В дальнейшем нам хотелось бы исключить посредничество программиста между графической системой и пользователем. Для этого необходимо разработать такой способ взаимодействия неподготовленного пользователя с системой, который давал бы ему возможность самостоятельно создавать компьютерные графические модели объектов для той или иной предметной области и задавать отношения между этими объектами. Программную систему такого уровня будем называть графической операционной средой, а язык, на котором осуществляется построение таких моделей, авторским языком. Авторский язык должен быть прежде всего доступным, т. е. овладение основными командами и конструкциями языка не должно требовать значительной программистской и математической эрудиции. Сокращения, аббревиатуры и графические обозначения должны быть простыми и естественными. Необходимо, чтобы авторский язык позволял пользователю работать на уровне моделей объектов, их свойств и отношений между ними, являясь тем самым естественным и удобным пользовательским интерфейсом графической операционной среды. Данные требования к авторскому языку относятся, как нам кажется, к более общему методологическому принципу проектирования программных средств, который можно было бы назвать принципом естественного взаимодействия. Этот принцип, на наш взгляд, распространяется на любые разработки программного обеспечения. Полиэкранный способ отображения графической информации как раз и позволяет организовывать наиболее естественное взаимодействие пользователя с графической операционной средой.

Желающие получить дополнительную информацию могут обратиться по адресу: 630098 Новосибирск, Приморская, 22.

А. БЛИШУН, М. СИМОНОВ, Г. ШАПИРО

Инструментальные программные средства формализации профессиональных знаний учителя

В области компьютеризации образования существенными являются три направления, требующие особых подходов, методик, программ.

1. Общеобразовательное направление, задачи которого решаются введением в средних школах и учебных заведениях курса «Основы информатики и вычислительной техники». Его цель определяется все большим проникновением информационной технологии в производство и общество в целом и заключается в том, чтобы дать каждому современному человеку представление о такой области знаний и деятельности, как информатика (в широком смысле этого слова). В настоящее время основным средством обработки информации является вычислительная техника. Это и определяет ее место и роль в данном курсе.

2. Подготовка учащихся к использованию вычислительной и микропроцессорной техники, или так называемая задача подготовки пользователей. Большинство учащихся по окончании учебных заведений придет на современное автоматизированное производство и им необходимо иметь навыки использования средств автоматизации. Такая задача может решаться как в рамках трудового обучения, так и в системе профессиональной ориентационной работы. Взаимосвязь первого и второго направлений должна дать максимальный эффект.

3. Использование средств вычислительной техники для интенсификации процесса обучения. Это, на наш взгляд, наименее исследованное направление, в частности задача разработки педагогических программных продуктов, и является основным предметом обсуждения в данной статье.

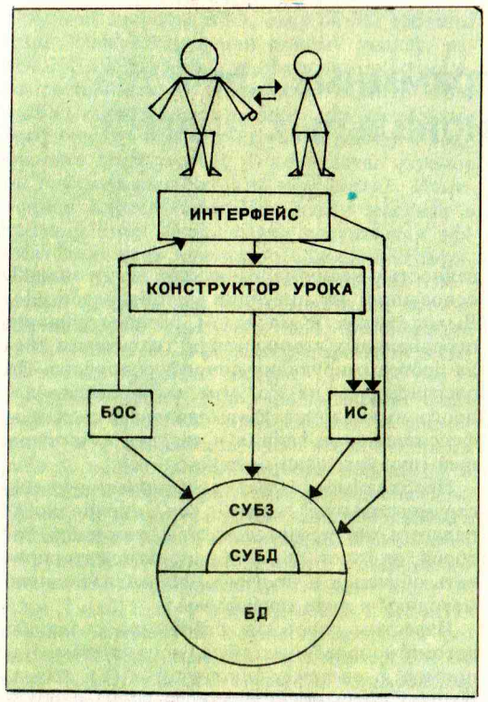
Многолетний опыт использования вычислительных систем показывает, что когда программист пытается формализовать задачи из новых для себя областей знания, то он сталкивается с большими трудностями. Подтверждается сказанное и мировым опытом разработки обучающих программных средств, свидетельствующим, что нередко затраты на производство программных продуктов не окупаются даже в малой степени педагогическим эффектом от их применения. По-видимому, в педагогике это связано с существенностью неформализованной части знаний, основанной на интуиции специалистов (1). Выход виден в массовом распространении персональных компьютеров, увеличении числа программирующих непрофессионалов. Но рассчитывать на то, что непрофессионал-программист может формализовать свои профессиональные знания и создать качественный продукт, пока не приходится.

Необходимо создать адаптированную инструментальную среду для специалиста-педагога, не программиста, с помощью которой он смог бы свободно автоматизировать обучение и получить формализованную методику в виде программы.

Известны советские работы по созданию автоматизированных обучающих систем. Например, в системе, описанной в (2), разработчику автоматизированного урока необходимо использовать специальный язык описания курса, не достаточно приспособленный к пользованию. В предлагаемой статье в системе закладывается максимальная адаптация к пользователю, реализуемая с помощью специального интерфейса, естественного диалога и обратной связи, а также инструментальных средств обработки текстов, графики и др.

АИС предназначена для создания автоматизированного урока (АУ). Такая среда создает условия и предоставляет средства для формализации знаний и приобретения навыков проектирования реальных автоматизированных уроков. Совокупность программ и наборов данных АИС представляет собой систему, открытую для встраивания новых компонент. Пользователи такой системы — учитель и ученик. Учитель работает в инструментальной среде с целью разработки автоматизированных средств обучения АУ. Он свободен при подготовке сценария и проведения АУ.

Рассмотрим общую структуру предлагаемой системы (рис. 1). Понятно, что она должна быть близкой к классу экспертных систем (4). Направление стрелки указывает на активность блока, относительно связанного с ним. В интерфейс входят средства, обеспечивающие адаптацию к проблемной области учителя. Основные части — лингви-



1

стический процессор, процедурный транслятор, средства передачи управления.

Лингвистический процессор выполняет такие функции:

знакомит пользователя с понятиями системы (терминами, командами, операциями, структурами);

позволяет вводить новые термины;

принимает с терминала и обрабатывает исходный текст на естественном языке.

В качестве интерпретации исходного текста генерируется последовательность некоторых формальных выражений. Результат интерпретации показывается пользователю, а в случае согласия эти выражения выполняются. Лингвистический процессор настроен на определенный словарь. Процедурный транслятор ориентирован на трансляцию принятой информации во входной язык конструктора уроков. Кроме того, интерфейс при приеме запроса пользователя производит передачу управления соответствующему блоку, который реализует этот запрос, выполняя функции, аналогичные функциям монитора. В интерфейс как составная часть входят аппаратно-программные средства, обеспечивающие ввод текстов и вывод реакции системы (звук, цвет, текст).

Второй, важнейший для пользователя, блок — «Конструктор урока». Он позволяет учителю формировать уроки на различных уровнях абстракций, проводить поиск аналогий среди уже имеющихся схем уроков в базе знаний, реализовать запросы, предоставляет консультативную помощь. Основные функции «конструктора» в формировании сценария и его наполнения — следующие:

выделение элементов урока (по указанию учителя);

определение условий;

формирование структуры урока;

выход на другой уровень абстракций (например, на нижний уровень, после заполнения содержанием элементов урока на верхнем уровне абстракций);

вход и выход в блок инструментальных средств (ИС) для наполнения содержанием элементов урока на самом нижнем уровне абстракций;

реализация запросов в исполнительские программы и распределение ресурсов системы (планирование вычислений).

Приведем фрагмент диалога с «конструктором урока» на примере проектирования АУ «План местности» из курса природоведения III класса.

Система (С). Работали ли вы в среде? Пользователь (П). Нет.

С. Хотите ли познакомиться с описанием имеющихся уроков?

П. Нет.

С. Будете ли проектировать ваш урок?

П. Да.

С. Давайте познакомимся, меня зовут Среда, а вас?

П. Сидор Никанорович.

С. А фамилия?

П. Панин.

С. Сидор Никанорович, начинаем проектировать урок. Описывать его будем с крупных (главных) разделов, постепенно разворачивая их до конкретных задач и примеров.

С. Назовите тему урока.

П. План. План местности.

С. Задайте основные разделы урока. Назовите их.

С. Первый раздел.

П. Оценка уровня знаний. Повторение понятий.

С. Будет еще раздел?

П. Да.

С. Назовите второй раздел.

П. Основной материал.

С. Будет еще раздел?

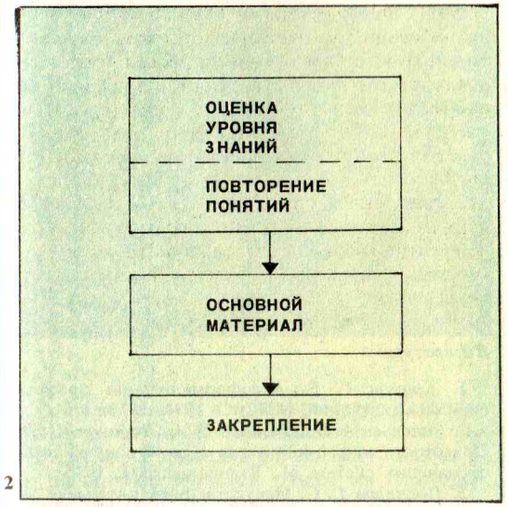
П. Да.

.

.

.

С. Уважаемый Сидор Никанорович, переходим к описанию последовательности выполнения разделов.



2

С. Последовательность в порядке ввода наименований разделов?

П. Да.

С. Ваша схема урока (рис. 2).

С. Будем наполнять содержанием разделы урока?

П. Да.

С. Будете пользоваться имеющимися разработками?

П. Нет.

С. Раскрываем первый раздел. Будете давать конкретный материал?

П. Нет.

С. Укажите основные элементы раздела 1 — «Оценка уровня знаний. Повторение понятий».

С. Назовите первый элемент.

П. Информация.

Аналогично разворачиваются остальные элементы раздела 1

С. Вы ввели следующие элементы...

С. Последовательность в порядке ввода наименований элементов?

П. Нет.

С. Номер элемента, выполняемого первым?

П. 1.

С. Условие перехода после элемента «информация» к следующим элементам есть?

П. Нет.

С. Следующий элемент.

П. 2.

С. Условие перехода после элемента «задание 1» к следующим элементам есть?

П. Нет.

С. Следующий элемент?

П. 3.

С. Условие перехода после элемента «проверка задания 1» к следующим элементам есть?

П. Да.

С. Номер элемента при выполнении условия?

П. 4.

С. Номер элемента при невыполнении условия?

П. 1.

.

.

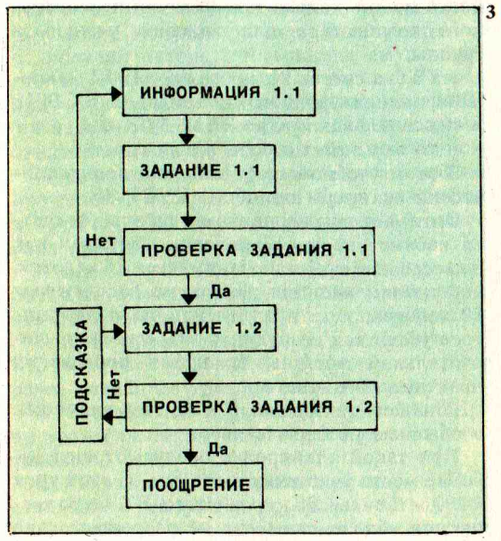
.

С. Сидор Никанорович, ваша схема раздела 1 (рис. 3).

На последнем, самом нижнем уровне, схемы будут представлять собой совокупность текстов, графиков, упражнений, которые наполняются с помощью блока ИС. Он содержит средства обработки текстов, реакции пользователя, графической информации, динамического моделирования, сервиса.

Все разработанное в «конструкторе урока» и блоке ИС хранится в базе знаний (3). Она состоит из системы управления базой знаний (СУБЗ), базой данных (СУБД) и непосредственно базы (БД). В нее можно попасть напрямую из интерфейса.

Последний блок системы — блок обратной связи (БОС), который обусловливает принадлежность среды к экспертным системам. С его помощью система накапливает информацию о пользователях, настраивается на известного пользователя, влияет на его



3

работу. Блок обратной связи накапливает и анализирует сведения о схемах запросов (диалога) пользователя, наиболее часто используемых им средствах блока ИС, его временных затратах.

В БОС имеются средства определения уровня компетентности пользователя (по отношению к среде), его ориентации. БОС анализирует параметр оригинальности, который выражается отношением количества самостоятельных разработок к общему числу используемых. Блок имеет систему оказания помощи пользователю при ориентации в среде, средства напоминания о санитарно-гигиенических нормах проведения автоматизированного урока.

Кроме того, в БОС включены средства поиска схем АУ, подобных проектируемой, и выдачи предложений пользователю. Заложена подсистема накопления сведений об ученике, его: времени усвоения материала, выполнения упражнений, реакции, уровня сложности решаемых им задач. БОС имеет подсистему тестов (для новичков).

БОС — единственный активный блок среды, т. е. при работе любых блоков системы он находится в активном состоянии, что реализуется, например, мультипрограммным режимом.

Необходимо отметить, что для полной реализации предложенной системы необходимы соответствующие ресурсы (быстродействие, объем памяти и т. д.), однако в рамках данной концепции возможна «усеченная» реализация среды на базе уже существующих классов учебной вычислительной техники.

Литература

1. *Громов Г. Р.* Автоформализация профессиональных знаний, МПСиС, 1986, № 3.
2. *Савельев А. Я., Новиков В. А., Лобанов Ю. И.* Подготовка информации для автоматизированных обучающих систем, М., Высшая школа, 1986.
3. *Поспелов Г. С.* Искусственный интеллект — основа новой информационной технологии. Международный симпозиум по искусственному интеллекту. Л., ИФАК, 1983.

Е. СЕРГЕЕВА
Преподаватель Минского политехникума

Первый опыт и наши проблемы

Как мы работаем в КВТ
Нерешенные вопросы

Обеспечить всеобщую компьютерную грамотность можно только при наличии персонального компьютера для каждого учащегося группы.

КУВТ, в состав которого входят 12 микро-ЭВМ индивидуального пользования БК-0010 и персональная микро-ЭВМ ДВК-2М, в какой-то мере дает такую возможность.

В этом учебном году Минский политехнический техникум приобрел КУВТ-86.

Оптимальным вариантом в школах является кабинет на 8—12 рабочих мест, так как в классе, как правило, занимается 25 человек, но группа учащихся техникума насчитывает 30 человек, т. е. при делении на подгруппы трое учащихся не будут иметь «своей» вычислительной машины во время проведения практического занятия.

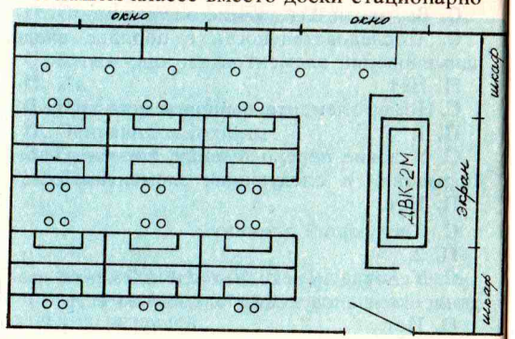
В нашем техникуме КУВТ был размещен в обычном учебном классе:

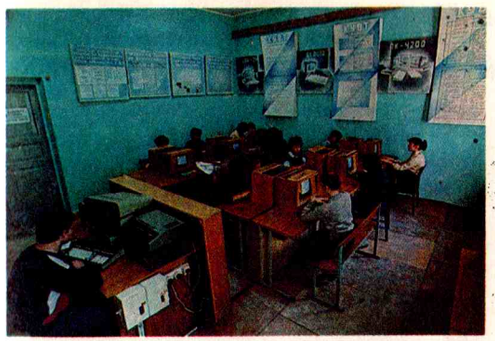
При такой планировке предусмотрены рабочие места учащихся, которым на этот урок «не достались» БК, кроме того, ВТ не размещается вблизи окон, что не разрешается по ТБ. В целях безопасности все БК были помещены в специальные ящички так, что у учащихся имеется доступ только к клавиатуре и экрану дисплея.

Как показывает опыт, чтобы не были нарушены санитарные нормы, сами БК лучше размещать не в центре стола, а на одной его половине, чтобы на другой учащийся мог вести необходимые записи.

Для удобства работы клавиатуру желательно «утопить», чтобы клавиши оказались на уровне стола учащегося.

В нашем классе вместо доски стационарно





установлен экран, а графопроектор устанавливается на столе преподавателя (использование мела приводит к большому количеству пыли, что является одним из факторов снижения работоспособности машин).

Занятие в основном планируется так, чтобы реализация программ по индивидуальным заданиям осуществлялась на последнем часу, т. е. последние 45 мин урока, либо при условии чередования составления программ и их реализации на ЭВМ составление программы по индивидуальным заданиям происходит при выключенных машинах.

В нашем классе размещены стенды с информацией, полезной при работе с машиной:

основные операторы языка Фокал;

основные директивы языка;

диагностические сообщения об ошибках.

Есть стенды со съемной информацией:

образцы оформления отчетов по практической работе в курсе ОИиВТ;

рекомендуемая литература по информатике;

ВТ в народном хозяйстве и др.

Преподаватель назначает трех консультантов из числа наиболее сильных учащихся, в обязанности которых входит оказание квалифицированной помощи учащемуся данной подгруппы. Каждый урок консультанты меняются. (Контролировать работу всех учащихся одному преподавателю сложно, особенно когда проходят первые уроки в классе и учащиеся еще не достаточно хорошо освоили клавиатуру БК). Мнение консультантов учитывается при выставлении итоговой оценки учащимся за практическую работу, что вносит элементы самоуправления, повышает интерес к предмету. Отношения между учащимися и преподавателем основаны на доверии.

Работу самих консультантов преподаватель оценивает с учетом помощи, оказанной ими другим учащимся, и по результатам проверки выполнения ими индивидуальных заданий.

Качественное своеобразие уроков, проводимых в классе ВТ, кстати сказать, не обязательно только практических, заключается в том, что преподаватель должен за время урока не только успеть опросить учащихся и объяснить новый материал, но также дать учащимся возможность реализовать хотя бы одну программу на БК. Преподаватель, естественно, испытывает определенный дефицит времени.

Возникает необходимость в программах-опросниках, выполнение которых идет не более 10—15 мин. Передав такие программы при помощи ДВК преподавателя на все БК, можно с их помощью опросить одновременно 12 человек, затем машина «сама» выставляет оценки. Такие программы содержат некоторый теоретический материал по конкретной теме, например: составление программ линейного типа на языке Фокал, составление программ разветвляющегося типа на языке Фокал, составление программ циклического типа на языке Фокал и другие; и вопросы по данной теме, отвечая на которые, учащийся в результате получит оценку. Если учащийся не получил пятерку, то машина ему «порекомендует», в какой книге и на какой странице ему можно найти нужный материал.

Создают эти программы учащиеся во время занятий кружка по программированию.

Кружковцами разрабатываются также программы, которые потом будут включены в качестве образцовых в раздаточный материал практического занятия.

К новому учебному году завершим разработку методических пособий для проведения практических занятий по следующим темам:

1. Инструкция оператора для работы с БК-0010. Составление и реализация программ линейного типа на языке Фокал.

2. Составление и реализация на БК-0010 программ разветвляющегося и циклического типов.

3. Работа с одномерными массивами. Составление и реализация программ на БК-0010.

4. Работа с одномерными массивами:

сортировка массива, замена элементов массива, поиск максимального и минимального элементов массива и их номеров и т. д.

5. Работа с двумерными массивами. Вложенные циклы. Составление и реализация программ на БК-0010.

6. Работа с массивами. Составление и реализация программ на БК-0010.

7. Графические возможности БК-0010. Построение графиков функций, рисунков.

8. Работа с системой ОС ДВК. Команды монитора. Работа со SCREENом. Создание файлов.

9. Составление и реализация программ на Бейсике линейного, разветвляющегося и циклического типов.

10. Работа с массивами. Составление и реализация программ для ДВК на Бейсике.

11. Графические возможности ДВК. Построение графиков функций.

Все практические занятия, как правило, будут содержать составление программ по тематике конкретной специальности, что дает учащимся увидеть реальную возможность использования ВТ в своей будущей работе.

Практический материал содержит информацию по Бейсику, по работе с ОС ДВК, так как в некоторых группах, где многие учащиеся занимаются в кружке программирования, мы решили в курсе ОИиВТ дать учащимся не один Фокал.

Это значительно расширяет кругозор учащихся, дает более глубокие знания.

Уроки, проводимые в классе, могут иметь самую разнообразную структуру, они не должны проводиться по шаблону.

К примеру, если преподаватель видит, что учащиеся справились с полученными заданиями, то на последних 10—15 мин урока учащимся можно разрешить поиграть. Мы иногда проводим дидактические игры «Эстафета», «Составь программу», «Найди ошибку» и т. д.

Группа учащихся разбивается на команды, в команде выбирается капитан. Всем командам даются одинаковые задания, которые необходимо реализовать на БК. Каждый участник команды имеет право выполнить одно лишь действие (например, набрать один оператор программы); если им была допущена ошибка, то следующий член команды имеет возможность только ее исправить. У команды есть возможность один раз за игру попросить помощь капитана.

Такого рода игры развивают чувство коллективизма, вызывают интерес к предмету, создают на уроке творческую атмосферу.

Возможности класса велики, велико и желание учащихся работать в нем, но нормальной работе, к сожалению, порой мешают частые отказы оборудования.

Необходимо воспитывать бережное отношение учащихся к вычислительной технике. В нашем техникуме ребята сами убирают кабинет, выполняют некоторые слесарные работы.

Теперь о проблемах. Хотелось бы поднять немаловажный вопрос о проверке тетрадей. Сколько преподавателю информатики приходится проверять работ по составлению учащимися алгоритмов при помощи алгоритмического языка, структурный схем, программ для ПМК и ЭВМ до того, как учащийся сможет реализовать свои программы на машине?!

Это занимает очень много сил и времени и, наверное, не меньше, чем у преподавателя математики. Поэтому, вероятно, необходимо решать вопрос, как платить за проверку тетрадей.

Имеется также множество проблем по организации кабинета.

Если техникум располагает одним КУВТом, то в этом классе на первом часу учебной пары может заниматься одна подгруппа учащихся данной группы, а на втором часу — другая с другим преподавателем. Конечно, начинать работу за пультом ЭВМ учащийся должен, имея уже готовый вариант программы, так как отведенного на работу времени может не хватить, а санитарные нормы работы учащихся с ЭВМ не позволяют продлевать эту работу на целый урок.

В связи с тем что учащимся необходимо после занятий выполнять домашнее задание, писать курсовые, дипломные работы, создавать обучающе-контролирующие программы и т. д., необходимо, чтобы в это время в КВТ находился лаборант (или оператор). Т. е. время работы класса должно продлеваться до 20—21 ч Кроме того, необходимо документально утвердить обязанности заведующего кабинетом, где бы точно было определено, за что он отвечает. К примеру, отвечает ли он за загрузку класса? Видимо, об этом должна заботиться администрация, а не заведующий кабинетом... Как будет производиться оплата за работу машин? В конце месяца? А если на протяжении месяца машина работала исправно, а перестала работать лишь в конце месяца, или наоборот, как раз в конце месяца ее отремонтировали? Т. е. в данном вопросе еще очень многое остается неясным.

Хотелось бы, чтобы для развития инициативы и интереса учащихся к созданию обучающих и контролирующих программ объявлялись соответствующие конкурсы. Не только для преподавателей ОИиВТ, но и для наиболее сильных, заинтересованных учащихся техникумов надо проводить научные конференции для обмена опытом.

В. КАЙМИН, Э. РУДАКОВ, А. ТИМОШЕНКОВ, А. ЩЕГОЛЕВ

Технология разработки учебных программных средств

Создание учебных программ, позволяющих решать значимые педагогические задачи, требует серьезных совместных усилий педагогов и программистов. Сложность состоит в том, что ППС должны удовлетворять, с одной стороны, психолого-педагогическим требованиям, а с другой — чисто техническим: надежности и эргономичности, гибкости и переносимости на другие вычислительные машины, и если технические требования носят безусловный характер, то психолого-педагогические требования должны быть, на наш взгляд, не менее строгими, поскольку от этого зависят и здоровье и духовное развитие наших детей.

Центральными в этом круге проблем являются определение педагогических целей в постановке обучения с применением учебных компьютеров и целостная концепция использования ЭВМ в учебной деятельности.

Первоочередной задачей, на наш взгляд, является разработка лабораторного практикума по информатике в соответствии с конкурсной программой по «Основам информатики и вычислительной техники. Цель практикума — научить ставить задачи, составлять формальные модели, описывать алгоритмы, составлять простейшие программы и интерпретировать результаты решений, получаемые с помощью ЭВМ.

Перечисленные умения в конечном счете опираются на навыки работы с ЭВМ в режиме диалога и тесно связаны с навыками чтения, письма, рисования, счета и поиска информации. Для их развития лучший из способов — работа с клавиатурным тренажером, учебным редактором текста, учебным графическим редактором, учебной базой данных и учебными электронными таблицами (экранным калькулятором). Конечная цель — в этой части практикума — сформировать умение интерпретировать результаты решения задач, используя перечисленные средства.

Следующий этап в лабораторном практикуме — формирование навыков составления алгоритмов на алгоритмическом языке и навыков составления, редактирования и испытания простейших программ на персональных ЭВМ. Основным средством для организации практикума может служить учебная система программирования (Бейсик, Паскаль или Рапира). Основным требованием к ней является наличие графических средств, средств обработки числовых и символьных данных, средств структуризации и редактирования программ.

Завершая практикум, можно использовать пакет демонстрационно-моделирующих программ, позволяющих в учебно-игровой форме познакомить учащихся с основными сферами применения ЭВМ в народном хозяйстве — работой со станками на производственном участке, в конструкторском бюро, в учреждениях, в лабораториях, в сфере обслуживания и дома. Конечной целью практикума является закрепление навыков работы с современными ЭВМ и формирование наглядных представлений об их возможном использовании в будущей работе.

Обучение умению составлять алгоритмы решения задач, записывать их в форме программ для ЭВМ, проверять их правильность, исправлять ошибки проводится у классной доски или в тетрадях, а на ЭВМ — ввод, испытание и редактирование программ. Проверка правильности алгоритмов и программ может и должна проводиться по отношению к постановкам задач и сценарию диалога с ЭВМ. Выработка умений постановок задач и описания необходимых для этого моделей, а также составления сценариев проводится в коллективных обсуждениях под руководством учителя и не требует пока специальных программных средств поддержки.

Учебные программные средства обработки информации и демонстрационно-моделирующие программы должны отвечать следующим общетехническим требованиям:

надежности — полному отсутствию сбоев и отказов как при правильных, так и при ошибочных действиях человека;

эргономичности — удобству и простоте правил работы с программой, «естественности» команд и учету эстетических факторов в оформлении графической информации;

гибкости — простоте внесения изменений в программу с целью модификации или расширения функций или информационной части программы;

мобильности — простоте переноса программ на другие ЭВМ.

Чтобы разработать программные средства для учебных ЭВМ, удовлетворяющие перечисленным и, кроме того, психолого-педагогическим требованиям, уточняющимся

в ходе педагогических экспериментов, необходимо выделить определенные этапы работы с регламентацией последовательности операций на каждом из них, а также процедуры контроля.

Существенным в обеспечении перечисленных качеств программ является то, что эргономичность, дружественность, надежность и гибкость — это во многом свойства функций программ, а не их внутренней структуры, и даже более того — сценариев, определяющих правила работы с ЭВМ и предоставления информации людям. По этой причине большую роль в создании качественных программ играет фаза проектирования, связанная с созданием, обсуждением и доработкой сценариев работы с ЭВМ, которые должны включаться в технические задания на разработку программных средств.

Надежность программных средств может быть обеспечена тщательным проектированием, включающим описание алгоритмов и функций, реализуемых в программах на инструментальных языках программирования; аттестацией алгоритмов и программ на их соответствие сценариям и внутренним спецификациям; массированными техническими испытаниями программ специалистами по тестированию.

Разработка программных средств учебного назначения во многом схожа с созданием продукции массового производства и должна проводиться в четыре этапа:

эскизное проектирование;
рабочее проектирование;
техническое проектирование;
опытная эксплуатация.

Результатом первого этапа должно быть создание эскизного образца программы и технического задания с обязательным описанием сценариев работы с ЭВМ. Составлению технического задания предшествует обсуждение сценария со специалистами, а утверждению технического задания — просмотр эскизного образца.

На втором этапе разрабатывается комплекс алгоритмов, обеспечивающих реализацию принятого сценария, и создается рабочий образец, который должен удовлетворять всем техническим требованиям и в точности соответствовать сценарию. Оценка рабочего образца проводится по результатам технических испытаний, которые могут выявить необходимость доработки сценария, комплекса алгоритмов или разработанных программ.

На третьем этапе разработанные программные средства дополняются полным комплектом документации, рекомендациями и методическими руководствами по применению программ.

На четвертом и наиболее ответственном этапе организуется педагогический эксперимент по использованию разработанных программных средств в учебных целях, в котором должны участвовать профессионалы — программисты и методисты, психологи и дизайнеры. Результатом педагогического эксперимента может быть переработка методических руководств и соответствующая модификация программных средств и проектной документации.

В научном плане наиболее важным результатом педагогических экспериментов должны стать уточненные требования к начальному уровню знаний, навыков и умений учащихся, уточнение учебно-педагогических целей и подтверждение психолого-педагогических требований и моделей. Исходные требования по этим позициям должны определять техническое задание на разработку учебных программных средств.

Главной частью технического задания на разработку диалоговых программных средств должен быть сценарий. Продуманность и детальность его описания служит гарантией успеха разработки. Все недочеты и недостатки сценария всплывут лишь на технических испытаниях либо в ходе педагогических экспериментов. Поэтому в обсуждении сценариев должны участвовать наиболее квалифицированные методисты и программисты, психологи и дизайнеры.

Сценарии программ для персональных ЭВМ с графическими средствами отображения информации — это последовательность сцен с описанием правил взаимодействия человека и ЭВМ. Сцена — это изображение относительно неизменной картины с описаниями движущихся объектов и текстовыми сообщениями, появляющимися на экране ЭВМ. Правила работы — это описания систем команд и запросов со стороны человека и наборов реакций ЭВМ в ответ на них.

Описание сценариев должно быть простым для чтения и понимания и в то же время достаточно строгим, чтобы давать однозначное толкование правил работы. Сценарии, с одной стороны, должны быть понятными непрограммистам, а с другой — служить эталонами для проверки правильности программ. Для описания сценариев необходима выработка полуформального языка, представляющего компромисс между функциональными спецификациями и неформальным словесным описанием, и в то же время обязательными компонентами сценариев должны быть основные графические решения в изображении и сцен, и составляющих их объектов.

Для описания комплексов алгоритмов можно и нужно использовать алгоритмический язык, изучаемый в настоящее время всеми учащимися средних школ и ПТУ, дополненный операциями ввода — вывода информации, в том числе операциями вывода графической информации.

Опыт применения излагаемой технологии проектирования программных средств для ПЭВМ в студенческом научно-производственном отряде МИЭМ показал, что до требуемого уровня доводились в срок те и только те программы, которые создавались на основе принципов структурного программирования и структурного проектирования с предварительным составлением сценариев и обсуждением их с педагогами и преподавателями. Применение ППС в учебном процессе дает существенное ускорение в изучении материала, приобретении опыта решения задач на ЭВМ и в составлении несложных программ на основе использования алгоритмического языка и навыков работы на ЭВМ с программами обработки информации.

Накопленный опыт привел авторов к убеждению, что компьютерные технологии обучения позволят существенно интенсифицировать процесс обучения.

От редакции.

С примером сценария можно познакомиться в молодежной рубрике нашего журнала в следующем номере.

Электронный кассир

На вокзале в Карл-Маркс-штадте около шести часов вечера работали всего две кассы. Очереди росли. До отхода поезда на Фрайберг оставалось около получаса.

Стоявший за мной молодой человек несколько раз посмотрел на часы, затем достал десятимарковую купюру и обратился ко мне: «Извините, не могли бы вы разменять на хартгельд?* Тогда я возьму билет без кассира».

На вокзалах многих городов ГДР можно приобрести билет с помощью компьютера. В больших настенных таблицах указаны алфавитные перечни крупных станций страны с четырехзначными индексами, которые являются также почтовыми индексами этих городов. Тут же несколько видеотерминалов, экранов, как у портативного телевизора, и возле каждого — клавиатура с цифрами. Есть и клавиша с обозначением «ошибка». Прикосновение пальца к «ошибке» или просто задержка на 15 секунд очередного ответа возвращают терминал в состояние первоначальной готовности к приему заказа.

...Молодой человек взял свой билет и торопливо направился к выходу на один из перронов. Я занял его место у терминала. На экране постепенно высвечивались слова: «Найдите в таблице и наберите код нужного вам места назначения».

Я заглянул в таблицу и набрал нужный индекс. На экране высветилось: 9200 — Фрайберг (Саксония). Затем появилось «меню»:

2 — нормальный билет
3 — детский
4 — для учащегося
5 — для делегированного**.

Я дотронулся до цифры «2». Экран высветил известные уже данные и спросил:

1 — туда
2 — обратно
3 — туда и обратно.

Я ответил «1».

Новое меню предложило выбрать:

2 — пассажирский
3 — скорый
4 — экспресс.

Для проезда в скором поезде или экспрессе предусмотрена доплата, которая при коротких расстояниях может оказаться даже дороже стоимости самого билета. Ближайший по расписанию поезд был пассажирским. Я прикоснулся к двойке.

Теперь компьютер «знал», какой билет мне нужен. В окошечке против надписи «нужно уплатить» появились красные цифры 3.40 (т. е. 3 марки 40 пфеннигов). Я опустил монеты в одну и две марки. Компьютер высветил цифры 0.40. Однако монет в 10 или 20 пфеннингов у меня не оказалось. Я бросил 50 пфеннигов и тотчас прочитал на экране: «Подождите несколько секунд! Ваш билет печатается... Берите!»

Одновременно с билетом в нижнее отделение автомата упала и десятипфенниговая монета — сдача.

...С каждым днем все больше пассажиров приобретают билеты «у компьютера» и все меньше — занимают очередь в кассу.

А. Яковлев

* Хартгельд («твердые деньги») — монеты.

** Для делегированных на длительное время в другой город предусмотрена возможность поездок на уикэнд со значительной скидкой к месту постоянного жительства.

Быстрее, проще...

С развитием электроэнергетики в количественном и качественном отношениях растет объем проверочных и профилактических работ. Особое внимание уделяется устройствам релейной защиты и автоматики (РЗА). Это естественно — от них во многом зависит надежность и безопасность эксплуатации электростанций, линий электропередачи и т. п. К сожалению, существующее испытательное оборудование скоро исчерпает свои возможности.

Применение тест-программ диагностирования и автоматизированных проверочных средств сэкономит время при проверках, повысит производительность труда обслуживающего персонала, снизит требования к его квалификации.

Использование в автоматических системах «жесткой» логики приводит к их специализации — в результате требуется много разных систем для проверки разных параметров. Если же комплектом оборудования управляет микро-ЭВМ, то его можно почти мгновенно переналадить для разнообразных испытаний.

Ждановский металлургический институт и Курсэнерго создали систему для проверки технического состояния РЗА на основе микропроцессора КР580ИК80А, позволяющую реализовывать самые сложные алгоритмы диагностирования. Будем надеяться, что выпуск ее не за горами.

В. ДРОЗДОВ
г. Рязань

О проверке математических формул с помощью калькулятора

В I части пробного учебного пособия по информатике на с. 83 говорится: «С помощью калькулятора нельзя доказывать теоремы. Например, нельзя доказать, что $\sin^2 x+\cos^2 x=1$. Для этого пришлось бы проверить тождество для всех значений x, которых существует бесконечно много». Это, безусловно, верно, однако приведенный пример неудачен: ведь, строго говоря, тождество $\sin^2 x+\cos^2 x=1$ нельзя проверить даже при нескольких значениях x из-за ограниченной точности вычислений (например, $\sin^2 10°+\cos^2 10°$ получается равным 0,999999966).

Поэтому лучше рассмотреть случай, допускающий однозначное вычисление на калькуляторе и к тому же не известный учащимся. Из множества таких примеров остановимся на следующем: найти натуральные n, при которых числа вида $3n-1$ будут квадратами натуральных чисел. Несмотря на коллективный поиск, дающий возможность за сравнительно небольшое время испытать много чисел, не встретится ни одного квадрата. Возникает гипотеза: числа вида $3n-1$ ни при каком n не являются квадратами натуральных чисел. Но доказать это предположение на калькуляторе невозможно ввиду бесконечности множества натуральных чисел.

Применим доказательство от противного. Пусть существует такое натуральное m, что

$$3n-1=m^2 \qquad (1)$$

Ясно, что m не делится на 3, тогда либо $m=3k+1$, либо $m=3k+2$, где k — натуральное число. В обоих случаях равенство (1) невозможно, ибо получается соответственно:
$n-3k^2-2k=2/3$ и $n-3k^2-4k-1=2/3$.

Я. МИРОЧНИК
Винницкий техникум электронных приборов

Сеть ДВК

Многие кабинеты информатики оснащены диалоговыми вычислительными комплексами ДВК. В комплект, поставляемый заводом-изготовителем, входит одна ЭВМ преподавателя (ДВК-2), оснащенная накопителем на гибких магнитных дисках (ГМД) и печатающим устройством, и 12—13 ЭВМ учащихся (ДВК-1). В ДВК-1 отсутствует накопитель на ГМД, а единственным языком программирования является сокращенная версия языка Бейсик, записанная в постоянное запоминающее устройство (ПЗУ). Связь между ЭВМ преподавателя и учащихся отсутствует. Такой класс можно использовать только в качестве тренажера при изучении языка Бейсик, так как при выключении питания программа пропадает, а единственное средство ввода — клавиатура. Эффективно использовать возможности ДВК-1 можно, только имея связь между ЭВМ преподавателя и учащихся. В этом случае есть возможность передать учащимся готовую программу и сохранить результаты их работы на диске. В журнале «Информатика и образование» (№ 1, 1986 г.) рассказывалось о классе из 12 ДВК-1 в школах Зеленограда. В этом классе ПЗУ с интерпретатором Бейсика заменено на ПЗУ с интерпретатором Фокала. Обмен возможен только на этом языке, что ограничивает возможности обучения. В Винницком техникуме электронных приборов создан класс, позволяющий подключить до 30 ДВК-1 к одной ДВК-2. ЭВМ преподавателя может работать в режимах передачи и приема. В ДВК-1 в ПЗУ сохранен интерпретатор языка Бейсик, но можно ПЗУ отключить и использовать другие языки. Для класса разработана сетевая программа, позволяющая работать в нескольких режимах.

1. Работа на языке Бейсик. Программа на Бейсике хранится на диске в ДВК-2 и сетевой программой передается в ДВК-1. После передачи учащиеся работают так, если бы программа вводилась с клавиатуры.

2. Работа на языке Фокал. Интерпретатор Бейсика отключается нажатием дополнительной клавиши на ДВК-1. С помощью сетевой программы в ДВК-1 передается интерпретатор Фокала. После этого можно набирать программу на Фокале с клавиатуры или ввести с диска ДВК-2. Можно использовать готовые прикладные программы.

3. Работа с программами в формате отображения памяти. Такие программы образуются при трансляции программ, написанных на языках Паскаль и макроассемблер, Фортран и др. Сетевая программа создает в ДВК-1 копию ОЗ ДВК-2. Затем программа работает в ДВК-1 так же, как в ДВК-2 после считывания с диска.

4. Режим копирования терминала преподавателя. В этом режиме преподаватель может демонстрировать учащимся любые тексты, набираемые с клавиатуры или считываемые с диска. Особенно этот режим полезен при изучении операционной системы. Учащиеся могут наблюдать реакцию ДВК-2 на команды оператора.

5. Режим приема. С помощью сетевой программы любой текст (программа на любом языке, или результат работы программы, или просто произвольный текст, набранный на дисплее) передается из ДВК-1 в ОЗУ ДВК-2 и на экран дисплея преподавателя. Учитель может этот текст записать на диск и распечатать.

6. Использование ДВК-1 как устройства подготовки программ. Например, учащиеся набирают программу на Паскале в ДВК-1, передают преподавателю в ДВК-2 для трансляции, а преподаватель программу в машинных кодах передает учащемуся.

Оборудование класса можно применять не только на уроках информатики, но и при изучении любых предметов. Возможность использования практически любого языка программирования облегчает обмен программами. Особенно эффективно использование класса при курсовом проектировании. До создания класса при выполнении расчетов на ДВК-1 учащиеся много времени тратили на ввод и отладку программы. Поэтому программы не могли быть достаточно сложными. С созданием класса имеется возможность разработать пакеты программ по всем основным расчетам, быстро загружать программу, предоставив учащимся возможность варьировать данные, вносить изменения, искать оптимальные варианты. Результат работы можно сохранить на диске до следующего занятия и распечатать. Опытная эксплуатация класса в техникуме показала его высокую эффективность при использовании в учебном процессе.

Таллинский научно-учебный центр представляет

Пакет прикладных программ автоматизации учебного процесса «Школьница»

ППП «Школьница» предназначен для оснащения школьных кабинетов, оборудованных вычислительной техникой. Предлагаемая реализация выполнена для ПЭВМ «Агат».

Класс задач, решаемых с помощью ППП «Школьница», включает в себя учебные задачи по программированию, учебно-производственную разработку программных систем учащимися и экспериментальную разработку программного обеспечения различных школьных дисциплин и административно-хозяйственных процессов (организация небольших информационно-поисковых систем для обработки данных об учащихся, оперативное подведение итогов школьных конкурсов и т. п.).

Основные пользователи «Школьницы» — преподаватели, стремящиеся повысить эффективность преподавания своего предмета с помощью новых средств, администрация школы, заинтересованная в более четкой организации школьного учебного процесса, учащиеся, осваивающие основы информатики и вычислительной техники и приобретающие в школе квалификацию программистов.

ППП «Школьница» используется автономно, без привлечения базового программного обеспечения ПЭВМ «Агат», реализована на ассемблере «Агата», поставляется на гибком магнитном диске (ГМД), на котором занимает 64К байт.

Для функционирования пакета требуется 96К байт оперативной памяти.

Исходными данными ППП «Школьница» являются программы на языках программирования Рапира и Робик, данные (числа, тексты, кортежи, записи, множества и файлы) и ответы на запросы системы «Школьница».

Пакет «Школьница» не требует программной среды. Включение его в какую-либо программную среду не предусмотрено.

Основное ограничение на применение пакета — сложность разработок, выполняемых с его помощью: он не предназначен для разработок большого объема.

От существующих программных средств, решающих аналогичные задачи, пакет «Школьница» отличается методической направленностью входных языков Рапира и Робик, разработанных специально для применения при обучении программированию. Кроме того, он содержит все необходимые в учебном процессе средства операционной системы и инструментальные средства редактирования текстов, работы с файлами на гибких магнитных дисках и отладки программ.

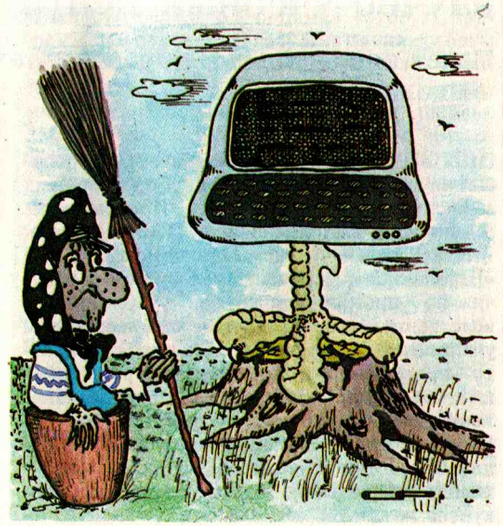
Минимальная конфигурация технических средств — ПЭВМ «Агат» с одним дисководом.

Ориентировочная стоимость — 100 руб.

Для заказа необходимо направить гарантийное письмо по адресу: 200026 ЭССР, г. Таллин, бульвар Кадака, 165.

За справками обращаться по телефону 68-26-46.

Веселый урок



Педагогический опыт

И. ЧЕРНИКОВА, О. ГОРБЕНКО

Подготовка и повышение квалификации учителей информатики

В последние годы в стране предприняты крупные меры для решения проблем, связанных с введением курса основ информатики и вычислительной техники,— подготовки преподавательских кадров и обеспечения учебного процесса вычислительной техникой. Работа проведена большая; говорить, однако, о значительных успехах в этом деле еще рано, и поэтому ни в одном из указанных выше направлений работа не должна быть ослаблена.

Дело подготовки учителей и преподавателей средних учебных заведений к ведению занятий по основам информатики было поручено ряду крупных высших учебных заведений страны. В Воронежской области — Госуниверситету, педагогическому и политехническому институтам в тесном контакте с областным институтом усовершенствования учителей. Их сотрудники приложили немало усилий, чтобы подготовиться и в предельно сжатые сроки провести занятия с учителями математики и физики, разработать и размножить методические рекомендации и пособия. Советом кабинета информатики областного института усовершенствования учителей (ИУУ) разработана система непрерывной методической помощи учителям информатики, которая успешно действует в течение последних трех лет. К работе привлекаются лучшие преподаватели университета. Она включает: ежемесячные семинары по программам IX и X классов для сельских и городских учителей, организацию региональных научно-практических конференций «Изучение основ информатики и вычислительной техники в школе и вузе», создание на Воронежском телевидении цикла передач «Компьютер в школьном классе», систематические консультации, экскурсии школьников в лаборатории вычислительной техники, подготовку методических рекомендаций и пособий по трудным вопросам школьной программы.

Вместе с тем организаторы этой работы отдают себе отчет в том, что уровень предметной и методической подготовки учителей информатики еще не соответствует требованиям дня. К сожалению, большинство учителей не могут систематически общаться с компьютером. Следует заметить также, что в зависимости от материальной базы обучение учащихся основам информатики осуществляется дифференцированно: во всех школах Воронежа и г. Борисоглебска Воронежской области преподавание предмета ведется по машинному варианту программы с использованием стандартных классов учебной вычислительной техники (КУВТ) и многотерминальных комплексов на базе микро-ЭВМ «Электроника-60» (базовых предприятий, УПК). В опорных школах области при проведении практических занятий используются программируемые микрокалькуляторы (ПМК). Да и объем курсовой подготовки не у всех учителей одинаков: в настоящее время в области есть учителя, прошедшие курсовую подготовку в объеме 144 ч, в объеме 72 ч есть и такие, которые в 1987 г. впервые приступят к преподаванию нового предмета.

Все это говорит о необходимости продолжения и совершенствования подготовки учителей. Учитывая существенные различия в уровнях теоретической подготовки учителей и их практических навыков работы с вычислительной техникой, совет кабинета информатики ИУУ принял решение о проведении летом 1987 г. курсов подго-

товки учителей по преподаванию основ информатики и вычислительной техники и повышения квалификации учителей, имеющих опыт преподавания этого предмета. Разработана такая структура курсов.

Двухнедельные курсы для учителей города и области, приступивших к преподаванию основ информатики год назад, и для учителей, впервые приступающих в 1987 г. к преподаванию этого предмета. Общий объем курсов — 72 ч. Основой программы этих курсов служит программа Минпроса СССР «Научно-методические основы информатики и электронно-вычислительной техники»; в программу включены также лекции по следующим вопросам:

Оборудование кабинета информатики и вычислительной техники	4 ч
Гигиенические нормы работы в кабинете ИВТ	2 ч
Анализ программы курса, методические рекомендации по каждому разделу с учетом опыта работы	6 ч

Недельные курсы общим объемом 36 ч для учителей Воронежа, преподававших основы информатики в течение двух лет. Здесь будут повышать квалификацию учителя, в достаточной степени овладевшие материалом школьной программы, поэтому в программу курсов включены темы, углубляющие и расширяющие полученные знания, а также вопросы, которые были предложены самими учителями:

Методические особенности курса и рекомендации по каждому разделу	4 ч
Алгоритмы сортировки и поиска информации	4 ч
Решение задач из курсов физики и математики	4 ч
Простейшие методы вычислительной математики	4 ч
Операционные системы: общая характеристика и описание конкретных ОС, входящих в программное обеспечение школьных ЭВМ	4 ч
Графические возможности компьютеров на примере языка	4 ч 4 ч
Практические занятия с использованием ПМК	4 ч 6 ч
Гигиенические нормы работы в кабинете информатики и вичислительной техники	2 ч

(Первая колонка цифр содержит объем лекционных часов, вторая — объем лабораторных часов.)

Недельные курсы общим объемом 36 ч для учителей опорных школ области и школ, имеющих комплект ПМК. Учитель опорной школы района является руководителем методического районного семинара учителей информатики, и те знания, которые он получает на курсах, будут переданы остальным учителям района по системе районной методической помощи. В программу этих курсов включены разделы:

методика преподавания отдельных тем, использование наглядных пособий	4 ч
алгоритмы сортировки и поиска информации	2 ч
практические занятия с использованием ПМК	— 14 ч
лабораторный практикум на ЭВМ	— 8 ч
решение задач из курсов математики и физики	4 ч 2 ч
гигиенические нормы работы в кабинете информатики и вычислительной техники	2 ч

Проводимые в этом году летние курсы для учителей и преподавателей основ информатики имеют одну важную особенность: впервые в их проведении будут принимать участие сами учителя из числа передовых педагогов, активно работающих над повышением своего теоретического уровня и над формированием эффективных методик преподавания нового предмета. Так, например, учителя школы № 86 Воронежа В. Б. Попов и Н. А. Тишукова разработали подробные поурочные методические рекомендации для проведения занятий по основам информатики с использованием ПМК. Они и их коллеги Квашин В. И. из школы № 82, Стародубцева Н. Ф. из школы № 77, Каширская Ю. В. из школы № 68, Власова А. С. из школы № 6 ознакомят с ними учителей города и области, проводя на курсах практические занятия по работе с ПМК.

А. ГРИНФЕЛДС, А. КАНГРО, Л. КУЗЬМИНА

Основы информатики в Латвийском университете

В Латвийском государственном университете им. П. Стучки создана кафедра основ информатики и технических средств обучения (ТСО) для преподавания «Основ информатики и вычислительной техники», «ЭВМ и программирования» студентам гуманитарных специальностей.

В 1986/87 учебном году все студенты дневного отделения Латвийского госуниверситета изучали дисциплину «Основы информатики и вычислительной техники». Для студентов гуманитарных специальностей продолжительность курса «Основ информатики и вычислительной техники» должна быть не менее 36 ч. В случае изучения основ информатики и вычислительной техники вместе с ТСО — 54 ч. Это предусмотрено проектом программы дисциплины «Вычислительная техника и ТСО» для педагогических институтов.

Цель нового курса — дать представление о роли ЭВМ в современном обществе, возможностях ее применения в учебном процессе, сформировать понятия о правилах и методах решения задач на ЭВМ.

В задачи курса входят:

ознакомление с ЭВМ на примере общих принципов работы;

овладение основными принципами алгоритмизации процессов;

формирование представления об этапах решения задач на ЭВМ;

знакомство с формами программированного обучения, одним из языков программирования и составление обучающих программ;

знакомство со сферой применения вычислительной техники, ее ролью в развитии современного общества.

Курс «Основы информатики и вычислительной техники» дает теоретическую подготовку по понятиям: информация — алгоритм — ЭВМ, формирует навыки алгоритмизации и программирования и закладывает основу для использования компьютеров в учебном процессе.

В лекционной части курса освещаются такие вопросы:

1. Информатика, ее предмет и актуальность.
2. Развитие вычислительной техники.
3. Персональная ЭВМ. Локальная сеть.
4. Структура и работа ЭВМ.
5. Основы алгоритмизации.
6. Общее представление о программах и программировании. Языки программирования.
7. Применение ЭВМ в учебном процессе. Автоматизированные обучающие системы. Формы программированного обучения.
8. Базы данных.
9. ЭВМ и общество.

На лабораторные работы отводится около 50—65 % учебного времени. Студенты учатся включать и выключать компьютер, пользоваться клавиатурой, вводить тексты в текстовом редакторе, записывать их на магнитную ленту. Главное же внимание на этих занятиях уделяется вопросам обучения с помощью ЭВМ. На примере обучающей системы «Рига» изучается ее текстовой язык и готовятся обучающие программы по специальности. Такая тематика занятий позволяет на практике продемонстрировать будущему учителю возможности использования ЭВМ в учебном процессе.

В специальной аудитории студенты работают с 11 микро-ЭВМ «Электроника БК-0010», соединенными между собой и с выходом на один магнитофон, к которому можно обратиться с любой ЭВМ, загрузить программу с магнитофона можно сразу на все компьютеры. Кроме того, занятия проводятся в классе Института физики твердого тела Латвийского госуниверситета, оборудованного серийно выпускаемым классом КУВТ-86.

Как программное обеспечение используется диалоговая обучающая система «Рига» и реализованный в ней Т-язык для программирования учебных материалов [1]. Она состоит из интерпретатора и редактора для ввода текстовой информации в ЭВМ. Возможности микро-ЭВМ «Электроника БК-0010» позволяют создавать программированные тексты, имеющие сложную структуру: синтезированный алфавит, музыкальные строки, речевые сообщения, программированные рисунки, а также программы на языке ЭВМ. В системе «Рига» преподавателями кафедры созданы программы для демонстрации возможностей ЭВМ, клавиатурный тренажер, дидактические игры для освоения клавиатуры ЭВМ, программа «Робот», моделирующая понятия:

алгоритм;

исполнитель алгоритма;

система команд исполнителя;

программная модель типовых структур данных.

Главная идея Т-языка состоит в том, что система воспринимает тексты на естественном языке и на национальных языках, а для гуманитарных специальностей это основная форма представления информации. Для создания учебных материалов более сложной структуры необходимо освоить инструкции Т-языка. 3—4 занятия со студентами требуются обычно на освоение основных инструкций, позволяющих писать несложные обучающие программы. На занятиях в течение семестра были рассмотрены такие типы обучающих программ:

программы-тренажеры для изучения правописания и перевода слов с английского, немецкого, французского языков на русский или латышский;

обучающие программированные тексты с разбором ошибок и многократным повторением вопроса;

контролирующие тексты, задача которых — выявить пробелы в знаниях школьника или студента, например по определенным разделам грамматики;

экзаменующие тексты для проверки знаний.

При изучении инструкций структуры текста студентам были предложены готовые фрагменты обучающих программ, составленных преподавателями. На этих примерах студенты быстрее усваивали инструкции языка, структуру организации обучающих программ. Затем они составляли сценарий выбранной обучающей программы, оформляли его на Т-языке и вводили в ЭВМ. Так разработано большое количество разных программированных материалов. Это хорошие примеры использования компьютера в преподавании русского языка, литературы, иностранных языков, истории, географии. Студенты этих специальностей с большим интересом относятся к изучению информатики и вычислительной техники. Некоторые из них приступили к выполнению курсовых и дипломных работ, используя обучающие программы.

На курсах повышения квалификации преподавателей гуманитарных специальностей университета и средних учебных заведений республики демонстрировались разработанные студентами и преподавателями обучающие программы, которые вызвали несомненный интерес.

Литература

Кузьмин Ю. Я. Т-язык и редактор информации системы «Рига». Рига: ЛГУ им. П. Стучки, 1986. С. 34

Н. ЛУНИНА

Что могут школьники

С 1981/82 учебного года в Научно-исследовательском вычислительном центре АН СССР (г. Пущино) ведется производственное обучение школьников по специальности «программирование». В результате обучения предусматривалось создание школьниками программного продукта, который может быть использован в производственной деятельности или учебном процессе. Это требовало серьезных теоретических и интенсивных практических занятий. За основу был взят адаптированный курс программирования А. Г. Кушниренко (механико-математический факультет МГУ)*. Вначале результаты обучения были неудовлетворительными. Во многом это объясняется тем, что школьники часто не хотели, а иногда были и не в состоянии усвоить трудный для них курс.

С переходом к формированию групп производственного обучения только из желающих ситуация изменилась к лучшему. Но и тогда многие школьники, столкнувшись с первыми трудностями, разочаровывались в выбранной специальности. В 1985/86 учебном году были созданы кружки по программированию, которые вели сотрудники НИВЦ АН СССР. Эти кружки служат базой для формирования групп производственного обучения. Предполагается, что на занятиях кружка учащийся должен убедиться в устойчивости своего интереса к программированию, а преподаватель — в способности ученика к этому виду деятельности.

В 1985/86 учебном году занятия кружков проводились в помещении НИВЦ АН СССР на ЭВМ СМ-4 с 7 дисплеями. Для обучения использовался язык МИНИ, разработанный на мехмате МГУ. В кружках занимались школьники V, VII и VIII классов. Работал также кружок для третьеклассников. Большие трудности возникали в связи с отсутствием специально оборудованного помещения, малым количеством дисплеев. Группы пришлось разбивать на подгруппы, каждый ученик занимался один раз в две недели по 2 часа.

Когда в экспериментальной средней школе г. Пущино появился кабинет учебной вычислительной техники «Ямаха», в нем стали заниматься две из четырех групп производственного обучения. В КУВТ были перенесены и занятия кружков, что позволило значительно увеличить количество кружковцев. Школьники разрабатывают программы по различным учебным дисциплинам: географии, астрономии, химии, истории, физике, русскому языку. Эти программы используются на занятиях кружков, имеются случаи их использования и на уроках.

С переходом на новую технику появились новые проблемы. Какой язык программирования выбрать? Какой должна быть программа курса обучения? Какие задачи использовать для практических занятий? Как, с чьей помощью преодолеть трудности первых занятий, возникающие у преподавателя при обучении большого количества новичков? Как часто проводить занятия и как долго они должны продолжаться? Следует ли ориентироваться только на программирование или только на работу с готовыми обучающими программами по различным учебным дисциплинам? При наличии этих двух видов работы, каким должно быть соотношение между ними?

При выборе языка программирования рассматривались две возможности: Е-практикум (машинная реализация учебного языка школьного курса ОИВТ) и программирование на Бейсике. Работа с другими языками программирования в условиях, когда 15 машин не имеют дисководов, пока невозможна. Предпочтение было отдано Бейсику по следующим причинам: наличие в Бейсике графических средств, что позволяет значительно расширить круг задач, доступных ученику средних классов; возможность создания на Бейсике, в отличие от Е-практикума, программного продукта. На первых занятиях новички знакомились с работой старшеклассников и видели, каких результатов можно достичь. Это служило дополнительным стимулом в обучении.

Что касается программы курса обучения, то в ее теоретической части решено придерживаться принятой схемы изложения материала, включающей:

понятие исполнителя;

управляющие конструкции;

технологию программирования «сверху-вниз»;

организацию данных.

Задачи для практических занятий разрабатывала автор данной статьи, руководствуясь следующими критериями: задачи должны быть доступны учащимся средних классов; способствовать усвоению изучаемых теоретических понятий; максимально использовать возможности ЭВМ. Вместе с кратким изложением теоретического материала они составили временное пособие, по которому проводились занятия.

Вопрос о помощи преподавателю решался следующим образом. На первые занятия за дисплеем приглашались сотрудники НИВЦ АН СССР. Теперь же в качестве помощников привлекаются старшеклассники из групп производственного обучения и кружковцы, это помогает новичкам быстрее освоиться и способствует возникновению контактов между старшими и младшими. Кроме того, старшеклассники участвуют в приеме зачета у новичков.

Из-за большого числа групп режим их работы несколько раз менялся. Наш опыт показывает, что проводить занятия реже чем по 2 часа в неделю неэффективно. Занятия с такой частотой могут приносить ощутимые результаты лишь при двух дополнительных условиях: наличия у каждого учащегося пособия; контактах со старшеклассниками, овладевшими курсом.

Необходимо решить еще один существенный вопрос: сколько времени следует отводить для собственно программирования, сколько — для работы с готовыми обучающими программами по различным школьным дисциплинам? А также: оправданы ли оба эти вида работ на занятиях или следует ограничиться одним из них? Должно ли их соотношение быть одинаковым для всей группы или индивидуальным? В настоящее время занятия проходят так: 1,5 часа — программирование, плюс 0,5 часа — работа с обучающими программами. Если ученик предпочитает и последние полчаса уделить программированию, то ему предоставляется такая возможность. Нужно отметить, что чем слабее группа, тем чаще предпочтение отдается второму виду работы.

Сейчас в кружках занимается около 70 человек. Количество занимающихся резко снизилось при попытке ввести домашние задания. Степень усвоения материала в группах очень разная. Можно говорить об усвоении всего курса лишь в одной из групп — восьмиклассников второго года обучения. Они приступили к самостоятельной разработке обучающих программ. Неожиданными оказались результаты в группе шестиклассников второго года обучения. Успешно усвоив все темы и научившись самостоятельно писать и отлаживать программы объемом полтора десятка строк, содержащие ветвления со сложными условиями, циклы, работу со строками и графикой, они, тем не менее, не смогли перейти к программам в несколько десятков строк. В связи с этим хотелось бы отметить один тревожный факт. Многие из этих ребят имеют доступ к компьютеру не только в кружке и привыкли смотреть на него как на источник удовольствия. Поэтому добиться от них выполнения какой-либо работы, не связанной с программированием простейших игр, очень трудно.

Автору совершенно очевидной представляется необходимость создания кружков, выполняющих разные функции: первоначального знакомства с компьютером и, условно говоря, «совершенствования мастерства». Кружков для начинающих должно быть много, для детей различного возраста. В идеале, у каждого школьника должна быть возможность попробовать себя в таком кружке. Тех ребят, у которых проявятся способности к программированию, следует объединять в кружки с другими требованиями, другими задачами. Вместе с учащимися из групп производственного обучения они могут работать над обучающими программами.

И, наконец, как занятия в группах производственного обучения и кружковая работа соотносятся со школьным курсом информатики? По мнению автора, такие формы обучения способствуют повышению успеваемости по школьному курсу ОИВТ.

В заключение хочется подчеркнуть, что достижение школьниками такого уровня в изучении программирования, который позволит им создавать программный продукт,— задача вполне реальная. На наш взгляд, для более эффективного достижения компьютерной грамотности необходимо развивать различные формы внеклассной работы.

* *Лунина Н. Л.* Обучение школьников программированию: Методические рекомендации / ОНТИ НЦБИ АН СССР. Пущино, 1986.

Летчик может быть спокоен

Когда в топливный бак попадает осколок, взрыв почти неизбежен — если не принять специальных мер. Для защиты самолетов и вертолетов в подобных случаях создана микропроцессорная система, добавляющая в бак по мере расходования топлива специальный негорючий газ так, чтобы его доля постоянно составляла 20 % содержащегося в баке воздуха.

Работает система следующим образом: из бака постоянно берутся пробы газовой смеси, поступающие на специальный датчик концентраций, вырабатывающий три сигнала, в которых кодируется плотность, температура и абсолютное давление пробы. На основе этих данных микропроцессор решает уравнение идеального газа $PV/T=K$ и вырабатывает аналоговый сигнал, пропорциональный разности между желаемой и фактической концентрациями. Управляемый этим сигналом электромагнитный клапан отмеряет необходимую добавочную порцию из баллона с негорючим газом. Летчик может быть спокоен.

Система такого типа хороша и для любого промышленного или химического производства, в котором поддержание заданного отношения газ/воздух имеет важное значение.

В. ПАТРИН
Директор Хабаровского индустриального техникума
М. ЖЕБРАК
Преподаватель Хабаровского индустриального техникума

Изучаем вычислительную технику

В условиях ускорения научно-технического прогресса средние специальные учебные заведения призваны содействовать реализации важнейшей государственной программы разработки, производства и использования электронно-вычислительной техники и автоматизированных систем. Путь к такому содействию лежит через внедрение вычислительной техники в учебный процесс. Преподавание нового предмета — основ информатики и вычислительной техники — только первый этап на этом пути.

На втором этапе резко возрастает интенсивность обучения, заинтересованность учащихся в приобретении новых знаний. Актуальные вопросы производства и использования вычислительной техники получат широкое отражение в тематике курсового и дипломного проектирования.

Все это приведет к тому, что на третьем этапе будут созданы объективные условия для непрерывного изучения и применения вычислительной техники в течение всего периода учебы в техникуме. Именно такая задача поставлена перед Хабаровским индустриальным техникумом, который определен базовым по изучению электронно-вычислительной техники и использованию ее в учебном процессе в системе Министерства промышленности строительных материалов СССР и Минвуза РСФСР.

Вместе с тем недостаточный уровень подготовки преподавателей различных дисциплин в области вычислительной техники является одним из сдерживающих факторов широкого внедрения ЭВМ в обучение. Ускорение этого процесса связано с овладением всеми преподавателями активными навыками работы на ЭВМ. С этой целью в 1985/86 учебном году для преподавателей техникума были организованы занятия по изучению электронно-вычислительной техники и использованию ее в учебном процессе, которые проводились в кабинете ВТ, оснащенном комплексом ЕС-7920. В его состав входит восемь дисплеев, устройство печати, групповое устройство управления и модем. Посредством стандартных устройств сопряжения комплекс связан с ЭВМ ЕС-1052 вычислительного центра Дальневосточного научного центра АН СССР.

В ходе занятий преподаватели познакомились с оборудованием КВТ, особенностями базового программного обеспечения, перспективами применения ЭВМ для оперативного управления учебным процессом.

В нынешнем учебном году занятия были продолжены по программам, разработанным Учебно-методическим управлением по среднему специальному образованию Минвуза СССР. Программа на 40 часов предназначена для преподавателей общественных и общеобразовательных предметов. В нее включены основные сведения об устройстве и принципах работы ЭВМ, понятия о программном обеспечении современных ЭВМ и автоматизированных обучающих системах.

Для преподавателей общетехнических и специальных предметов предназначена программа на 72 часа, дающая расширенное представление о микропроцессорной технике и микро-ЭВМ, автоматизированном проектировании, авоматизации технологических процессов, гибких производственных системах.

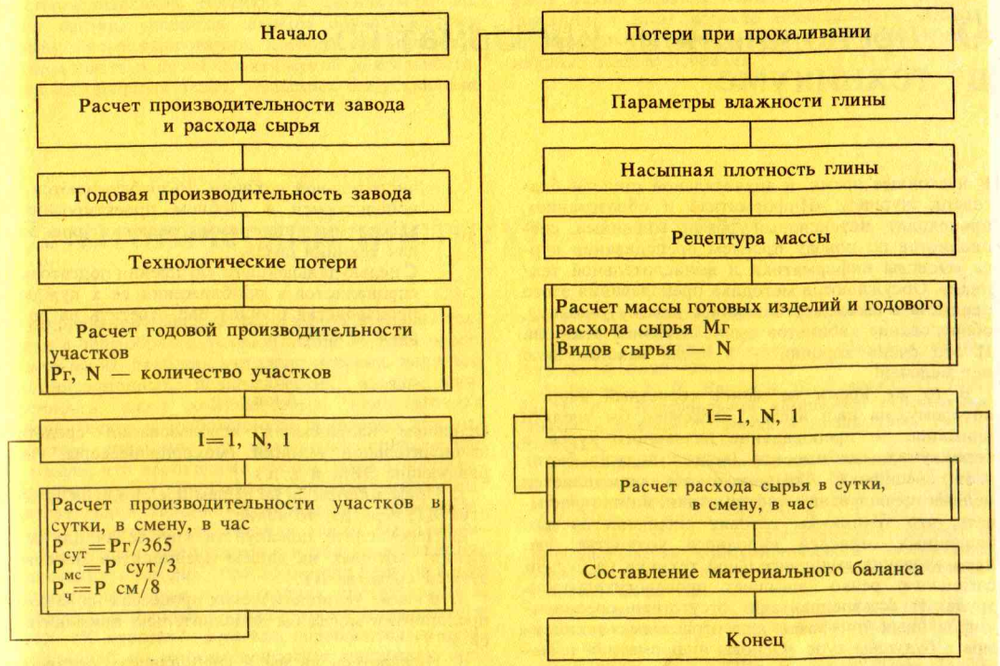
В нашем техникуме осуществлена подготовка преподавателей десяти техникумов отрасли и Хабаровска. Для групп обучаемых были разработаны соответствующие лабораторные практикумы, учитывающие специфику преподаваемых ими предметов.

Лабораторный практикум для преподавателей общетехнических предметов рассчитан на 40 часов и включает следующие работы:

1. Оборудование кабинета вычислительной техники. Правила техники безопасности при работе на ЭВМ — 2 часа
2. Знакомство с клавиатурой ЭВМ Начальные навыки работы на ЭВМ — 2 часа
3. Двоичное кодирование информации — 2 часа
4. Составление и исполнение алгоритма вычисления математической функции — 2 часа
5. Составление и исполнение алгоритма работы с массивами — 2 часа
6. Составление алгоритма по специальности — 12 часов
7. Построение и отладка линейной программы — 2 часа
8. Построение и отладка разветвляющейся программы — 2 часа
9. Построение и отладка циклической программы — 2 часа
10. Разработка программы по специальности — 6 часов
11. Разработка обучающих и контролирующих программ — 6 часов

Непросто складывались первые дни занятий. Наибольшие трудности в изучении вызвали специальные вопросы раздела «Принципы устройства и работы ЭВМ». Пришлось на ходу перестроиться, увеличить время на практические занятия. Раздел «Алгоритмы и алгоритмический язык» вызвал интерес преподавателей благодаря тщательному подбору примеров и задач. После выполнения практических работ по составлению и исполнению алгоритмов было предложено разработать алгоритм решения задачи из своей предметной области. Многие преподаватели взялись за составление алгоритмов расчетов, связанных с курсовым и дипломным проектированием. Некоторые заинтересовались возможностью автоматизировать при помощи ЭВМ процесс контроля знаний.

Алгоритм расчета



На занятиях установилась по-настоящему творческая атмосфера. После изучения основных конструкций языка Бейсик (на базовом ВЦ внедрена пользовательская процедура программирования на Бейсике) занятия были продолжены в кабинете вычислительной техники. Преподаватели довольно быстро освоили особенности работы на дисплее ЕС-7927, входящем в состав удаленного дисплейного комплекса, научились вводить и редактировать несложные программы и приступили к составлению программ по собственным алгоритмам.

В процессе работы у преподавателей возникали вопросы по поводу отсутствия на данной ЭВМ тех или иных аппаратных и программных средств, необходимых для оптимального решения выбранной задачи, что свидетельствует о глубоком понимании сущности разрабатываемой программы, реальной оценке ее возможностей и ограничений.

Большинство программ в ходе занятий были отлажены и доведены до уровня готового программного продукта. Некоторые из этих программ представляют несомненный интерес и на них стоит остановиться особо.

Преподаватель Спасского индустриального техникума Е. И. Хохлова разработала программу численного интегрирования. При изучении этого вопроса в курсе математики расчет выполняется вручную для относительно несложных примеров; кроме того, используются наиболее простые формулы интегрирования, точность которых невысока. Рассмотреть более сложные примеры преподаватель не имеет возможности, так как на вычисления уходит слишком много времени. Разработанная программа снимает вычислительные трудности и позволяет уделить внимание прикладному значению определенного интеграла.

Интересна программа расчета курсового проекта по технологии строительной керамики, разработанная преподавателем Хабаровского индустриального техникума Л. А. Остапенко. Алгоритм расчета приведен на рис. Работа организована в форме диалога учащегося с ЭВМ. По запросам программы он вводит необходимые данные, записывает промежуточные результаты, используемые затем при составлении материального баланса.

Заканчиваются занятия на курсах защитой рефератов по выбранной теме. Рефераты содержат описание задачи, методику расчета в терминах той предметной области, откуда задача взята, алгоритм решения и распечатку программы. Рефераты преподаватели увозят с собой, а в памяти ЭВМ остаются программы по математике, электротехнике, экономике и многим другим предметам, которые сейчас с успехом используются на уроках. Кроме практической ценности разработанные на занятиях программы дают огромный психологический заряд преподавателям. Опыт первых месяцев работы показывает, что они уезжают с курсов убежденными в возможности и необходимости применения вычислительной техники в учебном процессе, уверенными в своих силах. Уверенность эта базируется на прочных знаниях, подкреплена навыками работы на ЭВМ, что несомненно поможет преподавателям в поисках нетрадиционных путей и форм работы с учащимися.

Л. ПИСКАВЦОВА, В. БОЯРСКАЯ, Э. ЛИТВАК
Киевский политехникум связи

О преподавании информатики в техникуме

В настоящее время, в значительной степени благодаря журналу «Информатика и образование», происходит интенсивный обмен мнениями специалистов по поводу проблем преподавания курса «Основы информатики и вычислительной техники». Обсуждаются методика преподавания этого предмета в школе, требования к новому учебнику, оборудование кабинетов вычислительной техники. И это очень хорошо, — в столь важном деле нет мелочей!

В то же время не менее серьезный вопрос, которому, на наш взгляд, следовало бы уделять внимание — преподавание названного курса в техникумах, где имеется (вернее должна быть) своя специфика. Специфика эта определяется целями преподавания информатики. Важно учитывать, что техникумы готовят специалистов для различных отраслей народного хозяйства, где использование вычислительной техники не будучи самоцелью резко повышает производительность труда. К сожалению, при отсутствии специализированных программ для отраслевых техникумов в будущем курс «Основы информатики и вычислительной техники» может потерять значимость. Ведомственные специализированные журналы, очевидно, не будут, да и не должны восполнять нехватку информации по этому вопросу. Эту задачу мог бы взять на себя журнал «Информатика и образование», публикуя статьи об опыте различных техникумов.

Мы предлагаем основные результаты работы в этом направлении в Киевском политехникуме связи (КПТС). Курс «Основы информатики» в КПТС включает в себя работу на ВК М-6000, которой мы располагаем.

При участии учащихся разработаны и используются в курсовом и дипломном проектировании программы решения задач размещения телефонных станций и коммутационных устройств связи. Они опираются на оптимизационный алгоритм и позволяют получить значительный экономический эффект при проектировании реальных сетей связи, который также рассчитывается в дипломных проектах. Решена с ориентацией на программированный опрос задача расчета параметров аппаратуры передачи информации. В стадии разработки находится и ряд других задач техники связи. Сделаны первые шаги в создании отдельных подсистем АСУ техникумом.

С целью учета нужд производства при подготовке учащихся в техникуме подготовлено следующее информационное письмо:

Руководителю _____

В Киевском политехникуме связи в 1985—1987 учебных годах проводится расширенная подготовка учащихся в области вычислительной техники. Разрабатываются и используются в учебном проектировании алгоритмы и программы решения ряда задач техники связи.

С целью дальнейшего улучшения подготовки специалистов и приближения ее к нуждам производства просим Вас ответить на вопросы анкеты.

Анкета

Основные направления использования средств вычислительной техники (микропроцессоры, управляющие ЭВМ и т. д.)

1. Применяются ли на вашем предприятии средства ВТ, если да, то какие?
2. Перечислите, пожалуйста, круг задач, для решения которых на вашем предприятии используются средства ВТ.
3. В каких технологических процессах на вашем предприятии могла бы дополнительно применяться ВТ?
4. Нуждаетесь ли вы в специалистах среднего звена, получивших дополнительную подготовку в области ВТ?

Письмо отправляется на предприятия связи, которые являются возможными местами распределения учащихся.

Из-за ограниченного числа часов, отведенных курсу «Основы информатики и вычислительной техники», значительный объем сведений, необходимых для эффективной практической работы на производстве, приходится давать на занятиях кружка информатики.

Преподаватели техникума прослушали курс по основам составления алгоритмов и программ, что позволило использовать вычислительную технику на занятиях по различным предметам.

Практика показывает, что в ряде случаев большие ЭВМ использовать нецелесообразно. На практических занятиях часто применяются программируемые микрокалькуляторы. В техникуме создана библиотека прикладных программ для ПМК. При подготовке к практическому занятию учащийся может получить в библиотеке вместе с методическими указаниями к практическому занятию и прикладную программу. Ознакомление с ней до занятий дает возможность более эффективно использовать рабочее время урока и сосредоточить внимание учащихся на существе решаемых задач.

Мы считаем эту форму изучения основ информатики и вычислительной техники чрезвычайно важной, так как она побуждает учащихся смотреть на ЭВМ как на средство решения производственных задач и разрушает представление о существовании «ЭВМ ради ЭВМ».

Особо хотелось бы остановиться на языке программирования. И здесь полезно помнить о практической направленности курса «Основ информатики и вычислительной техники» в техникумах. На наш взгляд, учащиеся должны обучаться тому языку программирования, который им придется использовать в своей практической деятельности. Для инженерных задач, решаемых выпускниками нашего техникума, таким языком является Фортран. Возможно, в техникумах бухгалтерского профиля таким языком должен быть Кобол. Унификация в этом аспекте преподавания «Основ информатики и вычислительной техники» в техникумах нецелесообразна.

Педагогический институт — школе

Преподавание информатики и вычислительной техники в общеобразовательной школе — дело новое. Здесь много проблем: методологических, научных, организационных, педагогических. Малая оснащенность школ компьютерами, недостаточная методическая разработанность некоторых тем, отсутствие дидактических материалов привели нас к мысли, что необходимо организовать консультационный пункт при кафедре информатики и вычислительной техники Челябинского педагогического института, где учителя могли бы обмениваться опытом преподавания и получать квалифицированную помощь (преподаватели педвуза ведут занятия не только в аудиториях института, но и работают в школах — дают уроки информатики в IX—X классах). Учителя обращаются за консультацией по многим вопросам: машинная поддержка курса ОИВТ, клавиатура конкретных ЭВМ, языки программирования Фокал и Бейсик, работа со вспомогательными алгоритмами, использование микрокалькуляторов на уроках информатики, составление программ для программируемых микрокалькуляторов, варианты языка Бейсик для конкретных ЭВМ («Агат», ДЗ-28, ДВК-2М) и др.

С методической точки зрения оказались сложными первые уроки в X классах по теме «Основной алгоритм работы процессора». Учителя затрудняются в выборе и составлении заданий по темам курса, в подборе вариантов самостоятельных и контрольных работ.

В 1986/87 учебном году, учитывая малый опыт преподавания информатики в школах, интересы учителей, вопросы, с которыми приходилось иметь дело на консультациях, при кафедре был организован семинар «Дидактические проблемы преподавания курса основ информатики и вычислительной техники в школе». Приводим тематику семинара и список доступной для учителя литературы (он в основном содержит научно-методические и научно-популярные статьи, опубликованные за последние годы в журналах «Советская педагогика», «Информатика и образование», «Математика в школе», «Квант», «Наука и жизнь»).

Тема 1. XXVII съезд КПСС о развитии вычислительной техники. Перспективы применения ЭВМ в образовании.

1. Материалы XXVII съезда Коммунистической партии Советского Союза. М.: Политиздат, 1986.
2. *Александров Г. Н.* О закономерностях процесса обучения // Советская педагогика. 1986. № 3. С. 64.
3. *Бабанский Ю. К.* Совершенствовать методы педагогических исследований // Советская педагогика. 1986. № 3. С. 42—43.
4. *Бойко А.* ПЭВМ в школе // Наука и жизнь. 1985. № 10. С. 20—25.
5. *Велихов Е. П., Ершов А. П., Лавров С. С., Громов Г. Р.* Персональный компьютер: перспективы близкие и далекие // Наука и жизнь. 1985. № 10. С. 19—25.
6. *Высоцкая С.* Компьютерную технику — в школу // Советская педагогика. 1986. № 1. С. 139—141.
7. Вычислительные комплексы будущих поколений // Наука и жизнь. 1985. № 8. С. 15—19.
8. *Гапочка М. П.* Научно-технический прогресс и совершенствование коммунистического воспитания подрастающего поколения // Советская педагогика. 1986. № 1. С. 85—90.
9. *Гершунский Б. С.* Прогностический подход к компьютеризации // Советская педагогика. 1986. № 7. С. 43—48.
10. *Данилов И.* Система «Рига» обучает студентов // Наука и жизнь. 1985. № 3. С. 30—32.
11. *Кузнецов А. А., Бешенков С. А., Смекалин Д. О.* ЭВМ на уроках математики // Математика в школе. 1985. № 6. С. 44—46.
12. *Кузнецов А. А., Галагузова М. А., Игошев Б. М.* Овладение компьютерной грамотностью в процессе технического творчества // Советская педагогика. 1986. № 2. С. 29—32.
13. *Малкова Т. В., Монахов В. М.* Математическое моделирование — необходимый компонент современной подготовки школьника // Математика в школе. 1984. № 3. С. 46—50.
14. Научно-технический прогресс и школа // Советская педагогика. 1986. № 1. С. 12—15.
15. *Романов В. Н.* Экспериментальный курс «Прикладная математика и программирование» // Математика в школе. 1984. № 4. С. 38—42.
16. *Щербаков С. Г.* Решения XXVII съезда КПСС — в жизнь // Советская педагогика. 1986. № 3. С. 3—10.

Тема 2. Научно-методические проблемы школьного курса «Основы информатики и вычислительной техники»

1. *Боковнев О. А., Левицкий М. Л., Литвиненко Г. Н., Нурмухамедов Г. М.* ЭВМ в учебно-воспитательных учреждениях ГДР // Советская педагогика. 1986. № 2. С. 110—114.
2. *Велихов Е. П., Ершов А. П., Лавров С. С., Громов Г. Р.* Персональный компьютер: перспективы близкие и далекие // Наука и жизнь. 1985. № 10. С. 19—25.
3. *Добровольский Б. Ф.* Реформа школы — в действии // Советская педагогика. 1986. № 3. С. 18—21.
4. *Литвинова А. Н.* Микро-ЭВМ во Франции // Советская педагогика. 1986. № 2. С. 114—119.
5. *Щербаков С. Г.* Решения XXVII съезда КПСС в жизнь // Советская педагогика. 1986. № 3. С. 3—10.
6. *Юшков С., Андрофагин А.* Проблемы нового курса // Информатика и образование. 1986. № 3. С. 120—122.

Тема 3. Анализ содержания программы и учебных пособий школьного курса «Основы информатики и вычислительной техники».

1. *Данилов И.* Программируют школьники // Наука и жизнь. 1984. № 5. С. 92—96.
2. К началу обучения основам информатики и вычислительной техники // Математика в школе. 1985. № 4. С. 5—16.
3. К началу обучения основам информатики и вычислительной техники // Математика в школе. 1985. № 5. С. 35—48.
4. *Львов О. Г.* Проблемы большого дела (задачи компьютеризации) // Математика в школе. 1985. № 2. С. 31—33.
5. *Монахов В. М.* О содержании курса «Основы информатики и вычислительной техники» // Математика в школе. 1985. № 3. С. 7—12.
6. *Тесленко И. Ф., Распопов В. Б.,* Задачи компьютеризации // Математика в школе. 1985. № 2. С. 29—31.

Тема 4. Кабинет информатики и вычислительной техники в школе.

1. *Бойко А.* ПЭВМ в школе // Наука и жизнь. 1985. № 10. С. 20—25.
2. *Боканс Я., Грибустс А.* На базе БК-0010 // Информатика и образование. 1986. № 2. С. 84—86.
3. *Денисенко А.* Испытания КУВТ-86 // Информатика и образование. 1986. № 2. С. 69—74.
4. *Дрига И. И.* Оборудование для КВТ // Информатика и образование. 1986. № 2. С. 66—69.
5. *Дрига И. И.* Учебные кабинеты — перспективы совершенствования // Советская педагогика. 1986. № 6. С. 37—41.
6. Кабинет вычислительной техники всех типов средних учебных заведений (на базе персональных микро-ЭВМ) // Информатика и образование. 1986. № 3. С. 5—17.
7. *Шаповаленко С. Г.* Реформа школы в действии // Советская педагогика. 1986. № 2. С. 10—12.
8. *Леонова Л., Савватеева С.* Занятия в КВТ: влияние на организм // Информатика и образование. 1986. № 3. С. 82.
9. ЭВМ на любой вкус // Наука и жизнь. 1985. № 8. С. 16—19.

Тема 5. Алгоритмы, алгоритмический язык учебного пособия 9-го класса как основа для подготовки к изучению языков программирования.

1. *Абрамов С. А.* Самый знаменитый алгоритм // Квант. 1985. № 11. С. 44—46.
2. *Ершов А. П.* Алгоритмический язык // Квант. 1985. № 10. С. 42—45.
3. *Ершов А. П.* Алгоритмический язык // Наука и жизнь. 1985. № 11. С. 61—63.
4. *Ершов А. П.* Алгоритмический язык // Наука и жизнь. 1986. № 1. С. 138—141.
5. *Ершов А. П.* Компьютер — алгоритм — алгоритмический язык // Квант. 1985. № 9. С. 50—54.
6. *Ершов А. П.* На родине великого ученого // Квант. 1984. № 8. С. 29—31.
7. *Каймин В. А.* Решение задач и построение алгоритмов // Квант. 1986. № 10. С. 47—50.
8. *Криницкий Н. А.* Алгоритмы вокруг нас.— М., Наука. 1984. С. 224.
9. *Нагорный Н.* Понятие алгоритма // Наука и жизнь. 1985. № 1. С. 64—70.
10. *Поснова М. Ф., Поснов Н. Н., Вязовский И. К.* Школа начинающего программиста (занятие третье) // Наука и жизнь. 1985. № 8. С. 104—107.
11. *Усманов З. Д., Ходжиев И.* Ал-Хорезми // Квант. 1984. № 8. С. 27—29.
12. *Успенский В. А., Семенов А. Л.* Решимые и нерешимые алгоритмические проблемы // Квант. 1985. № 7. С. 9—15.

Тема 6. Использование языка программирования Бейсик в школьном курсе основ информатики и вычислительной техники.

1. *Антипов И. Н.* Программирование на Бейсике // Математика в школе. 1985. № 4. С. 48—52.
2. *Бояринов А.* Машинная поддержка курса // Информатика и образование. 1986, № 2. С. 81—83.
3. *Гоголев В.* Мою секретаршу зовут «Искра» // Наука и жизнь. 1986. № 8. С. 60—61.
4. *Данилов И.* Школа начинающего программиста (занятие десятое) // Наука и жизнь. 1986. № 9. С. 88—93.
5. *Дуванов А.* Какой язык лучше? // Информатика и образование. 1986. № 2. С. 95—96.
6. *Икаунисекс Е.* Основы языка Бейсик // Информатика и образование. 1986. № 3. С. 73—81.
7. *Соловьев В.* Как я заведовал складом химреактивов // Наука и жизнь. 1986. № 2. С. 106—107.
8. *Щеглов В.* Пятиминутки с «Искрой» // Наука и жизнь. 1986. № 4. С. 82.

Тема 7. Возможности языка программирования «Рапира».

1. *Звенигородский Г. А.* Первые уроки программирования.— М.: Наука. 1985.

Тема 8. Дидактические возможности использования микрокалькуляторов в учебном процессе.

1. *Адриан В.* Помог калькулятор // Наука и жизнь. 1986. № 6. С. 104.
2. *Вязовский И. К.* Школа начинающего программиста (занятие второе) // Наука и жизнь. 1985. № 7. С. 113—119).
3. *Ершов А. П.* Решение задач с применением программируемого микрокалькулятора «Электроника Б3-34» / Информатика и образование. 1986. № 3. С. 61—72.
4. *Зелинская Т. Я.* Об оптимальной модели микрокалькулятора для школы // Математика в школе. 1984. № 4. С. 50—51.
5. *Ионов Г. Н.* Микрокалькулятор как средство обучения и контроля // Математика в школе. 1984. № 3. С. 51—54.
5. *Комиссаров С. В.* Школа начинающего программиста (занятие пятое) // Наука и жизнь. 1985. № 11. С. 65—69.
7. *Поснова М. Ф., Поснов Н. Н., Леонтьев В. А., Бакунин П. Н.* Школа начинающего программиста (занятие шестое) // Наука и жизнь. 1985. № 9. С. 100—104.
8. *Привень А. И.* Не беда, что нет функций // Квант. 1985. № 6. С. 20—21.
9. *Приходько Б. К.* Решение уравнений на микрокалькуляторе // Квант. 1984. № 10. С. 14—17.
10. *Рождественский В. В., Хлебутин С. Г.* Программирование на микрокалькуляторе: ветвление и цикл. // Квант. 1986. № 3. С. 45—47.
11. *Славин Г.* Школа начинающего программиста (занятие первое) // Наука и жизнь. 1985. № 6. С. 33—37.
12. *Системский Л. Н.* Программирование на микрокалькуляторе: игры // Квант. 1986. № 6. С. 42—43.
13. *Трохименко Я.* Школа начинающего программиста (занятие девятое) // Наука и жизнь. 1986. № 7. С. 72—73.
14. *Шевченко Г. С.* Применение микрокалькулятора при изучении темы «Предел последовательности» // Математика в школе. 1984. № 3. С. 54—56.
15. *Штернберг Л. Ф.* Зачем микрокалькулятору стек // Квант. 1986. № 5. С. 39—43.
16. *Штернберг Л. Ф.* Программирование на микрокалькуляторе; простейшие программы // Квант. 1986. № 2. С. 46—48.

З. Ямалдинова,
канд. пед. наук,
Челябинск

Журнал в журнале
для школьников, студентов,
учащихся СПТУ и техникумов

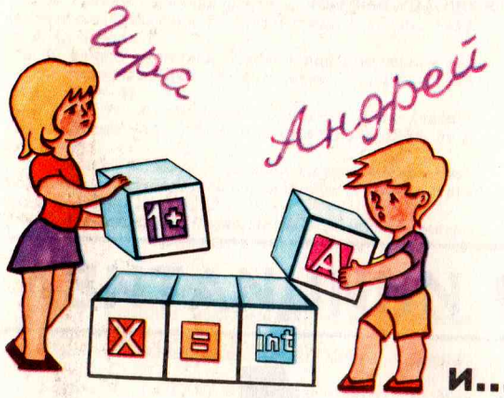
Издается при участии ЦК ВЛКСМ

МОЛОДЕЖНАЯ ИНИЦИАТИВА

4

Поскольку даже журнал «Веселые картинки» знакомит своих читателей с двоичной системой счисления, «Молодежная инициатива» сочла уместным опубликовать материал об обучении программированию детей младшего школьного возраста.

Надеемся, что эта статья поможет родителям любознательных и инициативных малышей организовать занятия в семье.



и... программирование

По профессии я программист. Мои дети — девятилетняя Ира и шестилетний Андрей — часто спрашивали, чем я занимаюсь на работе. Как объяснить им, что такое программирование? Рассказы о машинах, умеющих решать задачи, и о людях, которые их этому обучают, вызвали новые вопросы. Какие они — эти машины, какие задачи умеют решать? Программист — что-то вроде учителя, только не людей, а машин?

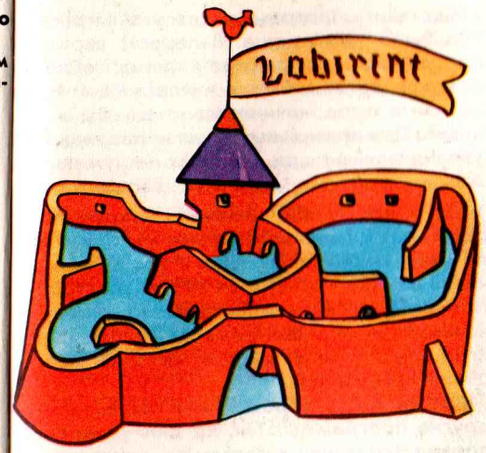
Я решил познакомить детей с настоящей вычислительной машиной, привел в зеленоградский клуб информатики, показал ДВК-2М и запустил несколько игровых программ. Дети сразу поняли: машина умеет играть! И играет очень хорошо! И опять вопросы: «А что у нее внутри? А как она играет?» Я рассказал, что внутри машины несколько микросхем (дети уже были знакомы с калькулятором, даже разбирали его). Кроме главной, называемой процессором, есть еще микросхема, которая работает с дисплеем, и другая, которая работает с диском, и еще другие. Я объяснял, а поток вопросов продолжал нарастать.

Когда в журнале «Веселые картинки» опубликовали описание двоичной системы счисления, и дети узнали, как считает вычислительная машина, они решили научиться считать так же. К моему удивлению, дочь и сын моментально освоили операцию перевода десятичных чисел в двоичные, причем самостоятельно экстраполировали правила перевода на трехзначные и даже четырехзначные числа. Затем мы начали играть в алгоритмы. Дети воспринимали их как лабиринт, по которому можно путешествовать. Я рисовал алгоритм и задавал какие-либо начальные значения. Если путь был найден правильно, то результат совпадал с ответом.

Удачные примеры игрового использования алгоритмов я нашел в газете «Пионерская правда», составил несколько предписаний, постепенно усложняя их. Все задания ребята выполняли с восторгом. Попутно освоились с понятием переменной и с основными конструкциями языков программирования.

В ноябре 1986 г. мы начали регулярные занятия на ЭВМ. Нас включили в расписание рабочей комнаты клуба информатики, и мы стали приходить туда по воскресеньям.

Первое время мы только играли. Но вот я предложил детям клавиатурный тренажер, на котором нужно нажимать соответствующие клавиши, а машина высвечивает на экране символы. Мы называли это игрой в буквы. На уровне игровой программы прошел и экранный редактор текстов. Здесь



детей больше всего восхитила легкость исправления текста (они уже были знакомы с печатающей машинкой, и сравнение было явно в пользу компьютера). Дети сами выдумывали различные игры с экранным редактором. Например, сначала вводили сплошной текст без пробелов, а затем возвращались и расставляли пробелы.

Постепенно запас игр начал иссякать, и тут я предложил научить машину новой игре. Это предложение вызвало восторг. Еще бы, ведь самим учиться не очень хочется, а учить машину — это совсем другое дело. Это гораздо интереснее.

Начали с игры «Отгадай число», в которую часто играли по дороге в детский сад или школу. Суть ее заключается в том, что один из участников загадывает число в диапазоне от 1 до 100, а другой пытается его отгадать. На каждую попытку второго игрока первый отвечает «больше» или «меньше» — и так до тех пор, пока ответ не сойдется.

Я предложил, чтобы одним из игроков была машина. Пусть, например, она загадывает число, а человек, играющий с машиной, будет его отгадывать. Сначала составили предписания, определяющие, что и в какой последовательности должна делать машина:

а) задумать число от 1 до 100;

б) сказать (для машины — написать на экране) фразу «Я ЗАДУМАЛА ЧИСЛО ОТ 1 ДО 100. ПОПРОБУЙТЕ ОТГАДАТЬ ЕГО»;

в) услышать ответ (для машины — принять то, что набрано на клавиатуре);

г) сравнить ответ с задуманным числом. Если задуманное число больше, то нужно написать «БОЛЬШЕ» и перейти к третьему шагу;

если задуманное число меньше, нужно написать «МЕНЬШЕ» и тоже перейти к третьему шагу;

если ответ правильный, нужно написать «ПРАВИЛЬНО» и перейти к началу игры (на первый шаг).

Записать эту последовательность действий, нарисовать схему не составило труда. Сложнее было с переводом на язык программирования. Пришлось объяснить, что, поскольку у машины нет глаз, прочитать наш алгоритм она не может. Чтобы задание было ей понятно, необходимо составить его на понятном машине языке — языке программирования. В качестве такого языка я выбрал Бейсик. Сначала ребятам не понравились английские слова PRINT, INPUT, IF, GO TO, но в конце концов они смирились, признав, что английский язык является международным для программистов. Чтобы ключевые слова и сама программа лучше запоминались, мы повесили распечатку на стену, и через несколько дней дети уже произносили эти слова свободно.

Начальный вариант программы «Отгадай число»

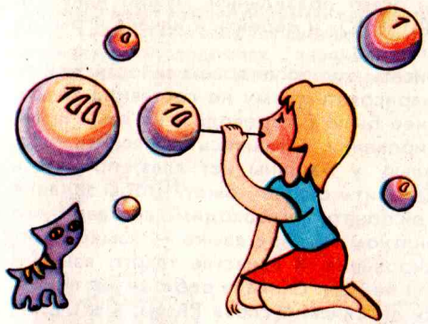
```
5 RANDOMIZE
10 X=INT (1+99*RND)
20 PRINT 'Я ЗАДУМАЛА ЧИСЛО ОТ 1 ДО 100'
25 PRINT 'ПОПРОБУЙТЕ ЕГО ОТГАДАТЬ'
30 INPUT A
40 IF X<A GO TO 200
50 IF X>A GO TO 300
60 PRINT 'ПРАВИЛЬНО'
70 GO TO 10
200 PRINT 'МЕНЬШЕ'
210 GO TO 30
300 PRINT 'БОЛЬШЕ'
310 GO TO 30
```

Очень скоро дети захотели усовершенствовать программу. Сначала ввели подсчет количества попыток. Затем дочка предложила, чтобы в конце игры машина задавала вопрос: «ПОИГРАЕМ ЕЩЕ?» По инициативе сына ввели печать «ОКРУЖЕНО» в том случае, когда введены ответы на единицу больше и на единицу меньше правильного. Мы также предусмотрели контроль адекватности задуманного числа заданному диапазону.

Заняв позицию консультанта, я стремился, чтобы ребята усовершенствовали программу самостоятельно.

Полный текст программы «Отгадай число»

```
1 RANDOMIZE
2 PRINT 'ВВЕДИТЕ НИЖНИЙ И ВЕРХНИЙ ПРЕДЕЛЫ'
3 PRINT 'ДЛЯ ЗАДУМАННОГО ЧИСЛА'
7 PRINT\ PRINT 'НАПРИМЕР: 1,100'
```




```
8 INPUT N1, N2
9 IF N1>=N2 THEN PRINT 'ОШИБКА'\ GO TO 8
10 X=INT(N1+(N2—N1)*RND)
20 K=0
30 Y1=N1
40 Y2=N2
50 PRINT 'Я ЗАДУМАЛА ЧИСЛО ОТ'; N1; 'ДО'; N2
55 PRINT 'ПОПРОБУЙТЕ ЕГО ОТГАДАТЬ'
60 INPUT A
73 IF A<N1 GO TO 400
77 IF A>N2 GO TO 450
78 K=K+1
80 IF X<A GO TO 200
90 IF X>A GO TO 300
100 PRINT 'ПРАВИЛЬНО'
110 GO TO 350
200 PRINT 'МЕНЬШЕ'
210 Y2=A
220 GO TO 320
300 PRINT 'БОЛЬШЕ'
310 Y1=A
320 IF Y2<>X+1 GO TO 60
330 IF Y1<>X—1 GO TO 60
340 PRINT 'ОКРУЖЕНО'
350 PRINT K; 'ПОПЫТОК. Я ЗАДУМАЛА ЧИСЛО'; X
360 PRINT 'ПОИГРАЕМ ЕЩЕ?'
370 INPUT T$
380 IF T$—'ДА' GO TO 8
390 STOP
400 PRINT 'ЧИСЛО ДОЛЖНО БЫТЬ БОЛЬШЕ ИЛИ РАВНО'; N1
410 GO TO 60
450 PRINT 'ЧИСЛО ДОЛЖНО БЫТЬ МЕНЬШЕ ИЛИ РАВНО'; N2
460 GO TO 60
```

Когда я почувствовал, что эта игра начала надоедать детям, предложил придумать другую. Идею следующей игры мы заимствовали из книги «Микрокалькуляторы в примерах и задачах». Суть игры в том, что на экране на короткое время высвечивается случайное число, которое затем необходимо набрать.

Составить программу для этой игры детям было уже проще. В первом варианте программы предусмотрен вопрос: «Сколько цифр должно быть в числе?» Во втором варианте игра начинается с однозначных чисел. При правильных ответах разрядность увеличивалась, при ошибках — уменьшалась.

Для того чтобы избежать сложностей, связанных с гашением экрана, мы применили следующий прием: до и после вывода числа на экран выводится 24 пустые строки. При этом число «пробегает» по экрану и исчезает за верхней рамкой. Это придало игре некоторый динамизм, который детям очень импонировал.

Кроме чисто игровых мы составляли и другие программы. Так, ко дню рождения дочери составили программу печати пригласительных билетов. Можно было бы решить эту задачу и средствами обычного редактора текстов, но тогда бы пришлось вписывать имена вручную или корректировать текст для каждого билета. Программа позволила формировать билеты с индивидуальными именами и соответствующими окончаниями прилагательных (дорогой Миша, дорогая Майя и т. д.).

Составили мы также простую программу для устного счета, работающую по следующей схеме: задумайте два числа от 1 до 9, умножьте первое из них на 5, прибавьте к произведению 7, удвойте полученную сумму, прибавьте к удвоенной сумме второе задуманное число, сообщите результат, и я назову вам задуманные числа.

К настоящему времени у нас сложился определенный режим занятий: в течение недели (обычно по пути в школу) мы обсуждаем сюжет следующей программы, затем, в субботу, составляем алгоритм и текст этой программы, а в воскресенье идем в клуб, где работаем на ЭВМ.

В дальнейшем мы намерены продолжать игры, постепенно усложняя программы и расширяя используемые средства: от диалоговых программ перейдем к экранным, затем начнем использовать графику и т. д. И никакой теории, никаких уроков, заданий, искусственных примеров (по крайней мере на ближайшие несколько лет).

На мой взгляд, обучение малышей программированию очень похоже на обучение их иностранному языку. Меня всегда поражало, с какой легкостью дети осваивают язык. Единственное, что для этого требуется,— это «окунуть» детей в иноязычную среду. И не нужны при этом никакие специальные занятия. Нужно только, чтобы язык стал для них естественным средством общения. Грамматика, синтаксис и т. п. понадобятся гораздо позднее. Так и в наших занятиях. Поскольку машина не понимает обычную речь, а потребность в общении с ней имеется, поневоле приходится осваивать язык, который она понимает.

Однако суть программирования не сводится к преодолению языкового барьера между человеком и машиной. Цель занятий с детьми я вижу не в том, чтобы изучить какой-либо язык программирования. Гораздо важнее, на мой взгляд, чтобы дети поняли: ЭВМ не более чем инструмент для решения практических задач, которым можно научиться управлять. Для младших школьников основной вид использования этого инструмента — игра и обучение. Занятия программированием способствуют развитию логики мышления, т. е. умению проанализировать задачу, разложив ее на элементарные условия, умению синтезировать решение в виде последовательности элементарных действий, умению четко и однозначно излагать свои мысли. Такие навыки необходимы не только в программировании, но и в любой другой деятельности.

Главная проблема, с которой я столкнулся,— это отсутствие методик обучения младших школьников программированию. Надеюсь, что мой небольшой опыт поможет родителям и педагогам найти правильный путь в подготовке малышей к получению «второй» грамотности.

А. ШАБАЛИН

Хотите разобраться?

На прилавке лежит книга — яркая привлекательная обложка, но несколько отпугивающее название: «Экспертные системы. Принципы работы и примеры»*. Интересно и, наверное, очень сложно... Рискнем заглянуть? Очень приятная компания авторов во главе с уже знакомым нам Р. Форсайтом дружелюбно вводит новичка в непростую теорию экспертных систем. Ясность изложения натолкнула на мысль: а не взять ли у книги «интервью» для «М. И.»? Сказано — сделано.

— Мистер Форсайт, что заставило вас заняться редактированием этого сборника?

— «Многие люди независимо от того, считают ли они себя специалистами в области компьютеров или нет, уверены, что в области информатики и вычислительной техники происходит что-то очень важное, и поэтому они хотят повысить свою подготовку настолько, чтобы быть в состоянии дать всему самостоятельную оценку, а если необходимо — и воспользоваться новыми методами.

Этому призвана помочь настоящая книга, в которой основные идеи экспертных систем объясняются для пользователей вычислительных машин, не знакомых с последними результатами научных исследований».

— Перед вами именно такой пользователь. Не могли бы вы дать свои объяснения в самых общих чертах?

— «Экспертная система рассматривается как результат создания в компьютере основанной на знаниях компоненты, соответствующей навыку эксперта, в такой форме, которая позволяет системе дать разумный совет или принять разумное решение о функции обработки данных. ...

Наиболее существенным в отношении экспертных систем является то, что они действуют чрезвычайно успешно. Уже существуют системы, превосходящие в медицинской диагностике искусных специалистов. То же самое наблюдается при интерпретации масс-спектрограмм, предсказании возможности заболевания зерновых культур, поиске полезных ископаемых и во многих других областях. Совершенно неожиданно люди и некоторые действительно крупные корпорации стали использовать программы искусственного интеллекта для своего обогащения. Экспертные системы наконец перечеркнули старое высказывание: «если нечто работает, то это уже не искусственный интеллект!»

— Компонента, основанная на знаниях... Я слышал о базах знаний, но чем они отличаются от баз данных, не понял.

— «База знаний содержит факты (или утверждения) и правила. Факты представляют собой краткосрочную информацию в том отношении, что они могут изменяться в ходе консультации. Правила представляют более долговременную информацию о том, как порождать новые факты или гипотезы из того, что сейчас известно.

*Экспертные системы. Принципы работы и примеры / Под ред. Р. Форсайта; Пер. с англ. С. И. Рудаковой. М.: Радио и связь. 1987.

Чем такой подход отличается от обычной методики использования базы данных? Основное различие состоит в том, что база знаний обладает большими творческими возможностями. Факты в базе данных обычно пассивны: они либо там есть, либо их нет. База знаний, с другой стороны, активно пытается пополнить недостающую информацию».

— *Однако база знаний выглядит опасно неустойчивой! Факты могут меняться, правила... Они могут оказаться и ошибочными!*

— «...создатели экспертных систем внесли новую идею в практику применения вычислительной техники: в системах, работа которых основывается на использовании знаний, не предполагается абсолютной безошибочности математического обеспечения. Удалите наугад одно правило из базы знаний экспертной системы — и качество ее работы (по всей вероятности) лишь слегка упадет. Удалите теперь наугад одно утверждение из обычной программы — и... страшно подумать, что произойдет!»

— *Но как ЭВМ может справиться с подобной неопределенностью?*

— «... существует... много способов работы с неуверенностью в данных! Имеются нечеткая логика, байесовская логика, коэффициенты уверенности, многозначная логика, если назвать только четыре из них. Были испробованы все виды схем, и, странное дело, большинство из них, по-видимому, работают.

Мое объяснение такого положения состоит в том, что организация знаний играет бо́льшую роль, чем связанные с ним числовые значения. В большинстве баз знаний предусмотрена избыточность, позволяющая экспертной системе прийти к правильному заключению несколькими различными маршрутами.»

— *Честно говоря, использование «нечеткой логики» внушает мне недоверие к правильности вывода. Мало ли что взбредет этой ЭВМ в... э-э... процессор!*

— «Одним из самых замечательных свойств, присущих классическим экспертным системам... является то внимание, которое было уделено интерфейсу с пользователем. В любой момент можно было спросить систему, почему была сделана такая дедукция или почему система задала такой вопрос пользователю. В системе, основанной на использовании правил, ответ обычно получается путем прослеживания еще раз тех шагов рассуждения, которые привели к данному вопросу или к данному заключению.»

— *Компьютер посоветует, если нужно — объяснит, почему совет именно таков... Похоже, в экспертных системах возникают зачатки личности?!*

— «... в программе EURISCO... используемый язык описаний (средство, в котором хранятся правила и понятия) оказался достаточно выразительным для того, чтобы в ней присутствовало в зачаточной форме самосознание, выражаемое «метаправилами». Это — система с высокой степенью интроспекции, тратящая массу времени на управление своим поведением, запоминающая обнаруженные ею правила и применяющая их к самой себе.

Один из открытых ею результатов характеризует некоторый иронический момент. Так, она обнаружила, что обычно правила, поставляемые человеком, оказываются лучше ее собственных, поэтому она создала эвристику:

ЕСЛИ правило создано машиной

ТО вычеркни его

К счастью, первым правилом, которое оказалось стертым системой EURISCO было именно это правило!»

— *Несмотря на все свои удивительные качества, экспертная система не является универсальным инструментом для решения любых задач. Когда ее применение эффективно?*

— «Ответ зависит от типа задачи, которую вы хотите решить. В табл. 2.1 представлен контрольный список характеристик степени пригодности подхода с использованием знаний. Если имеющееся вами в виду приложение больше относится к левой части таблицы, чем к правой, то следует всерьез рассмотреть перспективу использования экспертной системы».

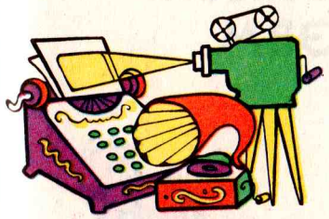
— *Спасибо за рассказ, мистер Форсайт. Я окончательно убедился, что вне зависимости от специальности в настоящем и планов на будущее стоит прочесть подготовленную вами книгу.*

К. ШЕХОВЦЕВ

Таблица 2.1

Подходит	Не подходит
Диагностика	Вычислительные задачи
Нет установившейся теории	Существует подходящая формула
Мало специалистов	Специалистов пруд пруди
Данные «зашумлены»	Известны точные факты

Не только ЭВМ



Нужен ли компьютер в детском саду? Вроде бы нужен, хотя пока неясно, как его там следует использовать, а главное — где взять.

Но электроника — это не только компьютеры. Французская группа фирм «Инфо» выпускает для самых маленьких полезную игрушку — книжки, картинки в которых нужно «рассортировать» по какому-то принципу, например: среди разных птиц найти водоплавающих. А правильность ответа проверяет «Бип-пен» — напоминающее карандаш устройство, в которое вмонтированы звуковой генератор, динамик, батарейки. Под рисунками специальной краской напечатана полоса, электрическое сопротивление которой может быть большим или малым. Коснувшись ее «Бип-пеном», ребенок узнает, точно ли он угадал: если да — электронная схема издаст приятный звук, если нет — неприятный.

Весьма просто и достаточно эффективно. Настолько просто, что подобные наборы для младших вполне можно делать в школьном радиоэлектронном кружке. И мы ждем от читателей «Молодежной инициативы» первых сообщений о ваших разработках для малышей. Прочитать подробнее о «Бип-пене» можно в журнале «Наука и жизнь» (1987, № 8).

Спаянный коллектив

(микродрама в системной магистрали)

Перевод с машинного

Действующие лица:

П. П. — Программа Пользователя
М. — Монитор
П. — Принтер
Т. К. — Телеграфный Канал
Т. — Терминал
Б. — Буфер

Приходит прерывание от Программы Пользователя.

П. П. Эй, Монитор!
М. Что?
П. П. Цифру «6» напечатать надо.
М. Эй, Принтер!
П. Я занят.
П. П. Тогда, Монитор, дай еще памяти.
М. Нет памяти, пиши на диск. Диск!

Диск поднимает флаг готовности.

Д. Вызывали?
М. Записать надо.
Д. Есть!
Т. К. Монитор! Для Вас получен байт.
М. Жди. Диск, записал?
Д. Нет свободного места.
М. Телеграфный канал! Давай свой байт.
Т. К. Уже пришел следующий, старый я стер.
М. Программа, это тебе прислали байт?
П. П. Нет.

Монитор засовывает байт в буфет.

М. Терминал! Пользователь что-нибудь нажимал?
Т. Нет.
М. Программа! Работай дальше.
П. П. Не могу, цифру «6» напечатать надо.
М. Принтер!
П. Я занят.
Т. К. Монитор! Для Вас получен байт.
М. Давай. Программа, это твой байт?
П. П. Нет.
М. Буфер!
Б. Я переполнен.

Монитор выбрасывает байт.

М. Принтер!
П. Свободен.
М. Печатай цифру «6».
П. Есть.
М. Программа, работай дальше.

Программа работает, проходит одна миллисекунда.

П. П. Эй, Монитор! Цифру «7» напечатать надо.

Газета «За науку» 10.10.86
(МФТИ)

Правила техники безопасности при работе с ЭВМ

1. Будьте внимательны, дисциплинированны, осторожны. Точно выполняйте указания учителя.

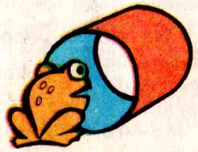
2. Не держите на рабочем месте предметы, не требующиеся при выполнении задания.

3. Перед выполнением работы внимательно изучите ход ее выполнения.

5. Включение ЭВМ производите последовательно.

4. Не включайте ЭВМ без разрешения учителя.

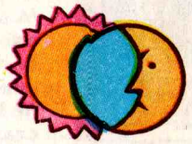
6. Не прикасайтесь к экрану дисплея, не трогайте провода.



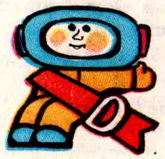
7. При работе с ЭВМ необходимо, чтобы экран находился на расстоянии 55—65 см от глаз, перпендикулярно линии взора.

8. Запрещается включать ЭВМ без крышки.

9. По окончании работы последовательно отключите ЭВМ.



10. Во время работы с ЭВМ запрещается перемещение по классу. Избегайте резких движений.



Составлено в НИИ школ Минпроса РСФСР С. КОЛОБОВЫМ

Внеклассная работа

Г. ГУТМАН, О. КАРПИЛОВА
г. Куйбышев

Азбука программирования

Изучение курса ОИВТ начинается в школе с девятого класса. Однако основные понятия программирования вполне доступны и более младшим школьникам. В кружке программирования можно в игровой форме познакомить с основными конструкциями алгоритмического языка, научить составлять простейшие алгоритмы и программы и подготовить ребят к пониманию принципов работы программируемых микрокалькуляторов и персональных компьютеров.

В настоящей статье предлагается методика безмашинного обучения основам информатики учащихся IV—VI классов.

Для знакомства с основными понятиями программирования используется простой исполнитель Муравей, впервые придуманный Г. А. Звенигородским.

Муравей может передвигаться по клеткам шахматной доски 8×8 и толкать перед собой один или несколько кубиков.

Таким образом, для проведения занятий потребуются шахматные доски и комплекты кубиков или картонных квадратиков разного цвета с буквами и цифрами. Один из кубиков (со звездочкой) — Муравей. Учителю для демонстрации работы исполнителя удобно использовать магнитную доску с соответствующим комплектом цифр.

Муравей выполняет любое из предписаний: ВВЕРХ К; ВНИЗ К; ВПРАВО К; ВЛЕВО К, указывающее ему, куда (вверх, вниз и т. д.) и на сколько клеток (К=0,1...,7) надо передвинуться. При К=0 Муравей остается на месте. Выйти за пределы доски или вытолкнуть туда кубики Муравей не может.

С помощью этого небольшого набора команд можно познакомить учащихся не только с простым исполнителем, но и ввести понятие линейной программы.

Линейная программа. Линейная программа — последовательность предписаний, ограниченная служебными словами **начало** и **конец.**

начало
последовательность предписаний
конец

Напишем программу для решения задачи.

Провести муравья по лабиринту (рис. 1) к выходу, набрав по дороге заданную сумму С=11.

Решение 1.

начало
ВНИЗ 1; ВЛЕВО 6; ВНИЗ 2; ВПРАВО 2; ВВЕРХ 1; ВПРАВО 1; ВНИЗ 4; ВЛЕВО 2; ВВЕРХ 2; ВЛЕВО 1; ВНИЗ 3
конец

Решение 2.

начало
ВНИЗ 6; ВЛЕВО 1; ВВЕРХ 3; ВЛЕВО 2; ВНИЗ 3; ВЛЕВО 1; ВВЕРХ 2; ВЛЕВО 2; ВНИЗ 3
конец

На этом примере школьники видят, что совершенно разные программы могут быть решением одной и той же задачи, т. е. правильность программы нельзя проверить, сравнив ее с «ответом». Для проверки составленной программы надо ее выполнить. Поставим Муравья (кубик со *) на шахматную доску с обозначенным на ней лабиринтом и будем двигать его, последовательно исполняя все написанные команды. Если при этом мы получим требуемый результат, значит программа составлена верно.

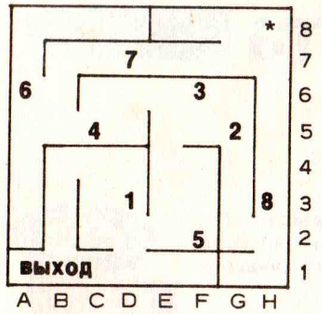
Предложенную задачу можно сформулировать в более общем виде: провести Муравья к выходу, набрав по дороге заданную сумму С от 10 до 28. Пусть на каждом поле доски Муравей может побывать не более одного раза. Какую сумму С тогда невозможно получить?

Много задач на составление линейных программ можно придумать, используя различные игры со словами.

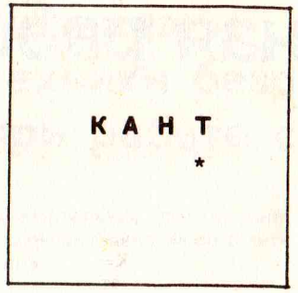
Превратить КАНТ (рис. 2) в ТАНК.

Решение.

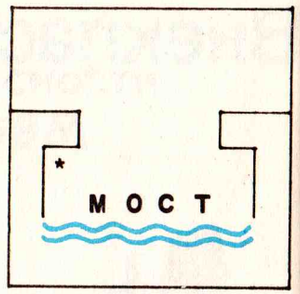
начало
ВВЕРХ 1; ВЛЕВО 1; ВВЕРХ 1; ВЛЕВО 3; ВВЕРХ 1; ВЛЕВО 1;
ВНИЗ 1; ВЛЕВО 1; ВНИЗ 2; ВПРАВО 3; ВНИЗ 1; ВПРАВО 1;
ВВЕРХ 1



1



2



3

конец

Обычно линейные программы довольно громоздки, и ребятам может стать скучно записывать все слова ВВЕРХ, ВНИЗ и т. д. В этом случае можно предложить заменять эти команды стрелочками: ↑K; ↓K; ←K; →K.

Циклы. После того как учащиеся освоили составление линейных программ, т. е. через 2—3 занятия, можно ввести новое понятие — цикл.

Простейший вид цикла — цикл «повторить». Его конструкция имеет такой вид:

повторить M **раз** — заголовок цикла
⟨серия предписаний⟩ — тело цикла
кц — конец цикла

Для иллюстрации этого понятия разберем решение следующей задачи.

Установить МОСТ (рис. 3) на опоры.

Р е ш е н и е. Запишем сначала линейную программу.

начало

ВНИЗ 2; ВПРАВО 1; ВВЕРХ 2;
ВНИЗ 2; ВПРАВО 1; ВВЕРХ 2;
ВНИЗ 2; ВПРАВО 1; ВВЕРХ 2;
ВНИЗ 2; ВПРАВО 1; ВВЕРХ 2

конец

Каждые три предписания, выполненные Муравьем, сдвигают очередную букву на 2 клетки вверх, 4 раза выполняется одна и та же серия команд. Поэтому естественно использовать цикл «повторить». Программа примет вид

начало
повторить 4 **раза**
ВНИЗ 2; ВПРАВО 1; ВВЕРХ 2
кц
конец

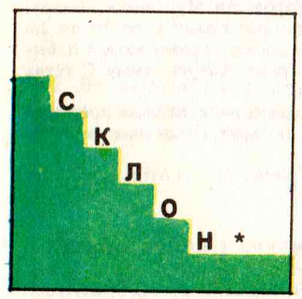
На рис. 3 Муравей первоначально стоит на поле B4. Как изменится программа, если Муравей сначала находится в другом месте, например, на левой опоре (на поле B6)? В этом случае тоже удобно воспользоваться циклом «повторить», но прежде придется сделать «вступительный ход», т. е. перевести Муравья на поле B4. Программа примет вид

начало
ВНИЗ 2
повторить 4 **раза**
ВНИЗ 2; ВПРАВО 1; ВВЕРХ 2
кц
конец

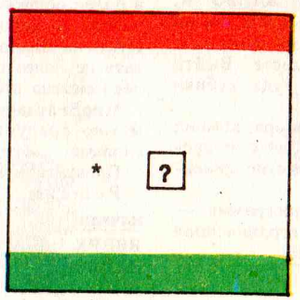
Если после установки моста Муравей должен уйти на правую опору, т. е. на поле G6, то после выполнения цикла «повторить» надо сделать еще несколько ходов.

начало
ВНИЗ 2
повторить 4 **раза**
ВНИЗ 2; ВПРАВО 1; ВВЕРХ 2
кц
ВПРАВО 1; ВВЕРХ 2
конец

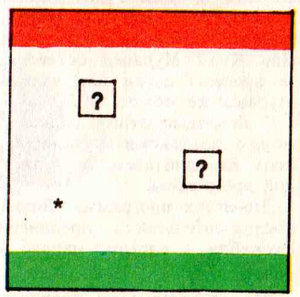
На примере последней программы школьники убеждаются в необходимости служебного слова



4



5



6

кц, которое отделяет тело цикла от продолжения программы.

На изучение цикла «повторить» требуется 1—2 занятия.

В программировании используются и другие конструкции циклов.

Рассмотрим цикл с параметром («для»). Его конструкция имеет следующий вид:

для K **от** $K_{нач}$ **до** $K_{кон}$ **шаг** Ш — заголовок цикла
выполнить
⟨серия предписаний⟩ — тело цикла
кц — конец цикла

Переменная величина K называется параметром цикла и принимает при первом выполнении тела цикла значение $K_{нач}$, при следующем — $K_{нач}$+Ш, затем $K_{нач}$+2Ш и т. д., пока не дойдет до последнего значения $K_{кон}$.

Разберем очередную задачу.

Разровнять СКЛОН (рис. 4).

Р е ш е н и е. Запишем сначала линейную программу.

начало

ВВЕРХ 1; ВЛЕВО 1; ВНИЗ 1;
ВВЕРХ 2; ВЛЕВО 1; ВНИЗ 2;
ВВЕРХ 3; ВЛЕВО 1; ВНИЗ 3;
ВВЕРХ 4; ВЛЕВО 1; ВНИЗ 4;
ВВЕРХ 5; ВЛЕВО 1; ВНИЗ 5

конец

Здесь каждая серия из трех команд перемещает вниз одну букву, каждый раз Муравей повторяет сходную последовательность действий, отличающуюся только числом клеток, на которое сдвигается очередная буква. Можно записать эту серию команд в общем виде: ВВЕРХ K; ВЛЕВО 1; ВНИЗ K, где K принимает значения 1, 2, 3, 4, 5.

Таким образом, для решения задачи удобно использовать цикл «для».

начало
для K **от** 1 **до** 5 **шаг** 1
выполнить
ВВЕРХ K; ВЛЕВО 1; ВНИЗ K
кц
конец

Заметим, что эту задачу можно решить и с помощью цикла «повторить»:

начало
повторить 5 **раз**
ВЕРХ 5; ВЛЕВО 1; ВНИЗ 5
кц
конец

но в этом случае Муравей проходит по доске более длинный путь и, следовательно, такая программа работает дольше.

Посмотрим, как изменится программа решения задачи, если первоначально Муравей находится на поле B7, а не на поле G2. Конечно, можно свести решение к предыдущему варианту, сделав до выполнения цикла «для» «вступительный ход» — ВПРАВО 5; ВНИЗ 5. Но удобнее сразу организовать цикл «для», однако теперь уже параметр цикла будет изменяться по-другому:

начало
для K **от** 5 **до** 1 **шаг** —1
выполнить
ВНИЗ K; ВВЕРХ K-1; ВПРАВО 1
кц
конец

Если в предыдущем случае на каждом шаге параметр K увеличивался на 1, то здесь он уменьшается на 1, т. е. принимает последовательно значения 5, 4, 3, 2, 1. На последнем шаге, когда K=1, Муравей сдвинется на одну клетку вниз, затем по команде ВВЕРХ K=1, т. е. ВВЕРХ 0, останется на месте, а по следующей команде ВПРАВО 1 сдвинется на поле G2.

На цикл «для» требуется также 1—2 занятия.

Ветвление. Для того чтобы познакомить школьников с понятием ветвления, надо научить Муравья еще некоторым действиям. Будем считать, что Муравей умеет различать цвет кубика, который стоит около него, т. е. умеет проверять условия типа КУБИК КРАСНЫЙ, КУБИК ЗЕЛЕНЫЙ, КУБИК ЖЕЛТЫЙ и т. д. Если рядом с Муравьем оказалось несколько кубиков, то в условии указывают, с какой стороны находится проверяемый кубик: СВЕРХУ, СНИЗУ, СЛЕВА, СПРАВА.

Когда учащиеся освоятся с понятием ветвления, можно будет рассматривать и составные условия типа КУБИК ЗЕЛЕНЫЙ **или** КРАСНЫЙ, КУБИК **не** КРАСНЫЙ и т. д.

Конструкция ветвления имеет вид

если ⟨условие⟩
то ⟨серия предписаний 1⟩
иначе ⟨серия предписаний 2⟩
все

Вторая серия предписаний вместе со словом **иначе** может и отсутствовать — это неполное ветвление.

Если условие выполняется, Муравей исполняет первую серию предписаний (ветвь «то»), если не выполняется — вторую серию (ветвь «иначе»). В случае неполного ветвления при невыполнении условия Муравей переходит к исполнению команд, стоящих после слова **все.**

Для иллюстрации понятия ветвления решим такую задачу.

Пусть нижняя горизонталь доски — зеленая («зеленая улица») (рис. 5), верхняя — красная («красная улица»). На поле E4 стоит красный или зеленый кубик. Попросим Муравья отвести кубик на «свою» улицу, т. е. на горизонталь того же цвета, что и кубик.

Как должен действовать Муравей? Ему надо подойти к кубику (ВПРАВО 1) и определить его цвет. Если кубик красный, Муравей должен сделать следующее: ВНИЗ 1; ВПРАВО 1; ВВЕРХ 4; если кубик зеленый, то ВВЕРХ 1; ВПРАВО 1; ВНИЗ 3. Запишем все это в виде программы.

начало
ВПРАВО 1
если КУБИК КРАСНЫЙ
то ВНИЗ 1; ВПРАВО 1; ВВЕРХ 4
иначе ВВЕРХ 1; ВПРАВО 1; ВНИЗ 3

```
все
конец
```

Поставим на доску вместо одного два кубика (рис. 6). Если заранее известно, что оба кубика одного цвета или кубики разных цветов, то задача не очень отличается от предыдущей, так как определив цвет одного кубика, мы узнаем цвет и другого.

Пусть, например, известно, что кубики имеют разный цвет. Тогда решение задачи таково:

```
начало
ВПРАВО 4
если КУБИК КРАСНЫЙ
  то ВВЕРХ 4; ВЛЕВО 3; ВНИЗ 5
  иначе ВЛЕВО 3; ВВЕРХ 4; ВПРАВО 3;
ВНИЗ 5
все
конец
```

Более сложной является ситуация, когда заранее про цвет стоящих на доске кубиков ничего не известно. В этом случае придется проверять оба кубика.

Подойдем к одному кубику: ВПРАВО 4. Если он красный, то ВВЕРХ 4 сдвигает его на нужное место, при этом Муравей оказывается на поле F7. Если кубик зеленый, то Муравей, выполняя ВПРАВО 1; ВВЕРХ 2; ВЛЕВО 1; ВНИЗ 3, сдвигает его на нижнюю горизонталь, а сам оказывается на поле F2. Теперь надо подойти ко второму кубику. Но в зависимости от положения Муравья это придется делать по-разному, т. е. дальнейший путь также зависит от цвета первого кубика, поэтому необходимые команды надо поместить в ветви «то» и «иначе» соответственно. При этом, чтобы было удобно организовать второе ветвление, Муравей в обоих случаях должен оказаться на одном и том же поле.

Таким образом, программа будет иметь вид

```
начало
ВПРАВО 4
если КУБИК КРАСНЫЙ
  то ВВЕРХ 4; ВЛЕВО 2; ВНИЗ 1
  иначе ВПРАВО 1; ВВЕРХ 2; ВЛЕВО 1; ВНИЗ 3;
  ВЛЕВО 2; ВВЕРХ 4
все
если КУБИК КРАСНЫЙ
  то ВНИЗ 1; ВЛЕВО 1; ВВЕРХ 2
  иначе ВВЕРХ 1; ВЛЕВО 1; ВНИЗ 5
все
конец
```

Здесь мы использовали два ветвления. Можно решить эту задачу и с помощью вложенных ветвлений.

```
начало
ВПРАВО 4
если КУБИК КРАСНЫЙ
  то ВВЕРХ 4; ВЛЕВО 3
    если КУБИК КРАСНЫЙ
      то ВЛЕВО 1; ВНИЗ 2; ВПРАВО 1; ВВЕРХ 2
      иначе ВНИЗ 5
    все
иначе ВПРАВО 1; ВВЕРХ 2; ВЛЕВО 1; ВНИЗ 3;
      ВЛЕВО 3; ВВЕРХ 3
если КУБИК КРАСНЫЙ
  то ВВЕРХ 2
  иначе ВЛЕВО 1; ВВЕРХ 2; ВПРАВО 1;
  ВНИЗ 5
    все
все
конец
```

Эта программа имеет уже более сложную логику, поэтому ее не стоит разбирать с самыми младшими учащимися.

При разборе программ, использующих ветвление, обязательно надо прослеживать на шахматной доске все возможные варианты исходной расстановки, чтобы убедиться в правильности программы.

Вложенные ветвления — довольно громоздкие конструкции, поэтому в ряде случаев вместо вложенных ветвлений удобнее использовать конструкцию «выбор». Эта конструкция имеет вид

```
выбор
  при ⟨условие 1⟩ : ⟨серия 1⟩
  при⟨условие 2⟩ : ⟨серия 2⟩
  . . . . . .
  при⟨условие K⟩ : ⟨серия K⟩
  иначе ⟨серия K+1⟩
все
```

Здесь (как и для ветвления) ветвь **иначе** не обязательна.

Исполняя эту конструкцию, Муравей проверяет последовательно каждое из условий и, как только обнаруживает верное, выполняет серию идущих за ним команд. На этом исполнение окончено.

Рассмотрим задачу.

Поставить кубик на полоску его цвета (рис. 7).

Решение. Поскольку здесь надо проверять много условий, удобно использовать **выбор.**

```
начало
  выбор
    при КУБИК КРАСНЫЙ : ВНИЗ 1; ВПРАВО 1; ВВЕРХ 3
    при КУБИК ЗЕЛЕНЫЙ : ВВЕРХ 1; ВПРАВО 1; ВНИЗ 4
    при КУБИК ЖЕЛТЫЙ : ВНИЗ 1; ВПРАВО 2; ВВЕРХ 1; ВЛЕВО 5
    при КУБИК СИНИЙ : ВПРАВО 2
  все
конец
```

Если заранее известно, что цвет кубика совпадает с цветом одной из полосок, то проверку последнего условия можно не проводить, заменив эту ветвь ветвью **иначе.**

Ветвление в цикле. Рассмотрим такую задачу.

На доске стоят красные и зеленые кубики (рис. 8). Развести их по «улицам» своего цвета.

Здесь Муравей должен определить цвет первого кубика и отвести его на нужную горизонталь, т. е. выполнить ветвление

```
если КУБИК КРАСНЫЙ
  то ВНИЗ 1; ВПРАВО 1; ВВЕРХ 4
  иначе ВВЕРХ 1; ВПРАВО 1; ВНИЗ 3
все
```

После этого Муравей должен подойти к следующему кубику. Куда идти, ВНИЗ 3 или ВВЕРХ 2, зависит от цвета первого кубика, поэтому эти команды надо добавить в ветви **то** и **иначе.** Оказавшись на поле C4, Муравей должен определить цвет следующего кубика и отвести его на нужную горизонталь, т. е. повторить последовательность тех же действий. Поэтому здесь удобно использовать цикл «повторить».

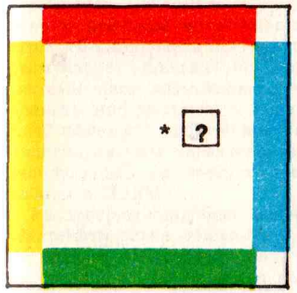
```
начало
  повторить 4 раза
    если КУБИК КРАСНЫЙ
      то ВНИЗ 1; ВПРАВО 1; ВВЕРХ 4; ВНИЗ 3
      иначе ВВЕРХ 1; ВПРАВО 1; ВНИЗ 3;
      ВВЕРХ 2
    все
  кц
конец
```

Пусть теперь исходное расположение кубиков иное (рис. 9). В этом случае удобнее использовать цикл «для»:

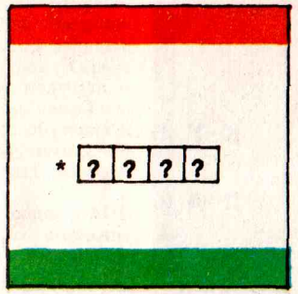
```
начало
  для K от 5 до 2 шаг —1
  выполнить
    если КУБИК КРАСНЫЙ
      то ВНИЗ 1; ВПРАВО 1; ВВЕРХ K;
      ВНИЗ K-2
      иначе ВВЕРХ 1; ВПРАВО 1; ВНИЗ 7-K;
      ВВЕРХ 7-K
    все
  кц
конец
```

В этой конструкции следует обратить внимание на организацию работы ветви **иначе.** Так как параметр цикла уменьшается, а число клеток, на которое надо передвинуть зеленый кубик на каждом шаге, увеличивается, то для вычисления длины пути приходится не прибавлять, а вычитать значение K. Этот прием также часто встречается в программировании, поэтому знакомство с ним полезно для школьников.

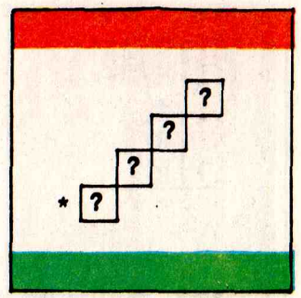
В приложении читатель найдет ряд задач, которые можно решать различными способами, используя рассмотренные выше конструкции.



7



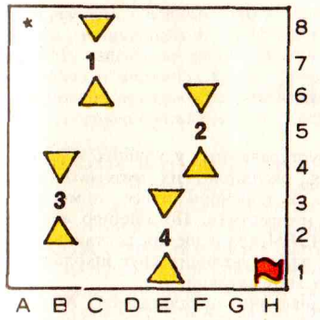
8



9

Приложение

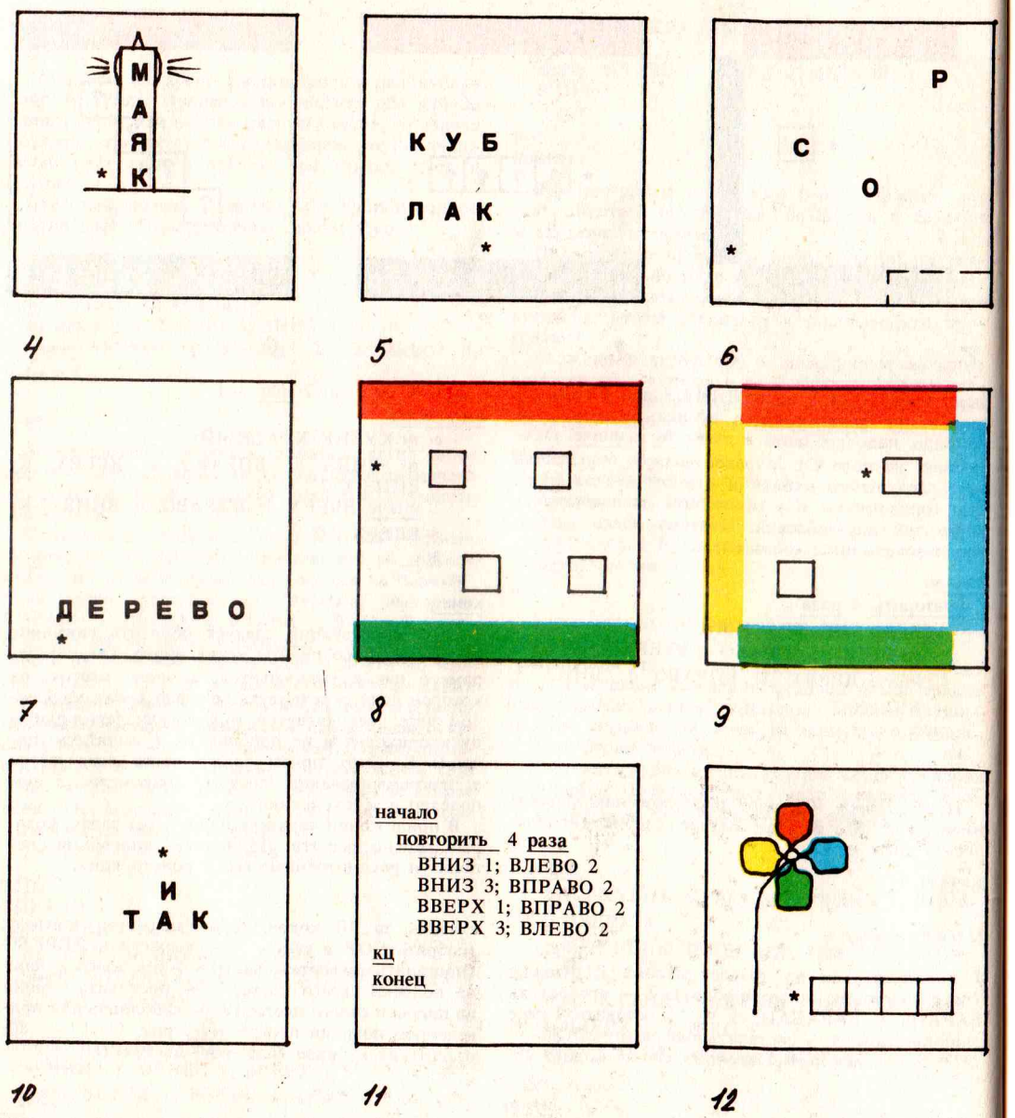
1 — спуститься по трассе слалома к флажку (по возрастанию номеров ворот); 2 — превратить БАРАНа в БАРАБАН; 3 — составить самое длинное слово; 4 — составить анаграмму МАЯК — ????; 5 — какое пятибуквенное слово можно составить за 10 ходов; 6 — подмести комнату (собрать СОР в углу); 7 — вырастить ДЕРЕВО (поставить его вертикально); 8 — поставить кубики на полоски своего цвета; 9 — поставить кубики на полоски своего цвета; 10 — выполнить для приведенной позиции программу с рис. 11; 12 — составить из кубиков «аленький цветочек».



1

2

3



Е. БАРАНОВА, И. ЕЛИЗАРОВА, Н. САФИУЛИНА

АОС «Математическая смекалка»

Разработанная на кафедре информатики ЛГПИ им. А. И. Герцена автоматизированная обучающая система[1] предназначена для решения с помощью ПЭВМ нестандартных задач по школьной математике. Формирование навыков решения таких задач способствует развитию у учащихся логического мышления, расширяет их математический кругозор, а это, в конечном итоге, помогает в усвоении курса математики. По мнению авторов, использование ПЭВМ для решения нестандартных математических задач увеличит круг школьников, интересующихся математикой.

Основное назначение предлагаемой АОС — обучить школьников методам решения задач определенного типа. При ее разработке авторы стремились максимально использовать графические и цветовые возможности ПЭВМ. Введение игрового компонента повышает интерес к задачам, стимулирует процесс поиска решения. Сервис системы дает возможность учителю получать и хранить информацию о ходе работы учащихся. АОС позволяет адаптировать процесс обучения к индивидуальным характеристикам обучаемых. Ее эксплуатация не требует специальных навыков работы с ПЭВМ.

Рассмотрим подробнее задачи, входящие в АОС «Математическая смекалка». Система содержит 24 задачи[2] (VIII серий по 3 задачи в каждой), рассчитанные на учащихся разного возраста. Задачи в сериях расположены по возрастающей степени трудности. Навыки, приобретенные при решении первой задачи серии, позволяют более рационально решить следующую, выработать определенный подход к решению аналогичных, а также способствуют формированию общего подхода к решению нестандартных задач.

Задачи разбиты на четыре группы. *Первая* содержит задания, в которых при определенных, постепенно усложняющихся условиях требуется расположить фигуры (звездочки) на поле 4×4. Эта группа задач, предназначенная для младших школьников, способствует формированию умения видеть сравнительно небольшую часть плоскости и располагать на ней фигуры в соответствии с условием задания (см., например, рис. 1). Программа позволяет произвольно ставить и убирать фигуры на поле в поисках правильного решения.

Работая с задачами этой группы, учащийся знакомится с клавиатурой ПЭВМ, получает представление о некоторых возможностях вычислительной машины.

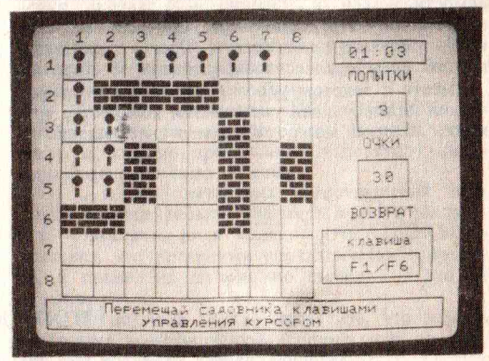
Вторая группа включает задания на выбор наиболее рациональных путей продвижения (различные лабиринты), предполагающие использование ситуаций, описанных в условии задачи, простейших арифметических фактов (см., например, рис. 2).

Задачи этой группы позволяют продемонстрировать преимущества ПЭВМ в поиске пути решения. Программа дает возможность вернуться назад, выбрать новый путь. Школьники учатся прогнозировать последствия выполненного шага.

Третью группу составляют задачи, в которых требуется представить данное число в виде суммы однозначных натуральных чисел (см., например, рис. 3). Это наиболее многочисленная группа, объединяющая три серии. Компьютер как средство обучения позволяет сделать условие задачи более интересным, увлекательным. Для этого, в частности, используются графические возможности ПЭВМ: слагаемые размещаются в вершинах, на сторонах и медианах треугольников, на диаметрах и в центре окружностей, на сторонах и в вершинах многоугольников и т. д. Схематическое изоб-

1

2а



2б

3

[1] АОС реализована на КУВТ «Ямаха» и может быть использована как на базовом, так и на инструментальном КУВТ.

[2] Бо́льшая часть задач взята из кн.: *Кордемский Б. А.* Математическая смекалка. М., 1963. Подбор задач, объединение их в серии, определение возможных путей решения и характера подсказок осуществлялись в соответствии с основными методическими принципами обучения решению задач. Работа велась под руководством профессора кафедры методики преподавания математики ЛГПИ им. А. И. Герцена И. В. Барановой.

4

ражение условия задачи с помощью геометрических фигур делает его более наглядным, облегчает поиск пути решения. Начиная с проб и ошибок, учащиеся постепенно овладевают определенным приемом решения подобных задач (несмотря на усложнение заданий, времени на решение затрачивается меньше).

Если на определенном этапе решения встретились затруднения, учащийся может использовать «подсказки», предусмотренные программой.

Задачи *четвертой* группы аналогичны задачам первой, но фигуры необходимо расположить на поле 8×8 (шахматная доска) (см., например, рис. 4). Это более сложные задачи, предполагающие наличие у учащихся некоторых навыков анализа последовательности шагов, их взаимозависимости.

В случае затруднений предусмотрена возможность получить «подсказку». Среди задач этой группы есть и классическая задача «8 ферзей».

Вместе с текстом условия задачи и ее графической иллюстрацией указывается количество времени, которое можно потратить на решение, а также число попыток, которые можно осуществить, сообщается о возможности «подсказки». Имеется также указание о том, как вызвать «подсказку». В конце работы ученику сообщается эмоционально окрашенная оценка его работы: «молодец», «надо было еще подумать», «ну, ничего, в следующий раз обязательно все должно получиться» и т. п.

Учителю система передает следующую информацию:

фамилия ученика, решающего задачу;

время, которое ученик затратил на решение;

количество попыток или перестановок, выполненных учащимися при решении;

количество очков, которое получил ученик за решение задачи;

оценка решения задачи.

Перечислим основные обслуживающие функции АОС. Сервис системы позволяет: автоматизировать процесс распределения задач между учениками (указанная учителем задача выбирается из набора программ и передается по сети на машину ученика); просмотреть на мониторе условия задач, включенных в систему, и распечатать их; распечатать инструкцию по работе с системой; получить информацию о классах, которые работали с системой (как на мониторе, так и на принтере); удалить устаревшую информацию, заменить ее новой; получить полную информацию о последнем занятии класса.

Таким образом, представляемая АОС обеспечивает индивидуальный подход в обучении. Режим сохранения информации предоставляет учителю возможность контролировать работу учеников и определять для каждого из них оптимальную последовательность предлагаемых для решения типов задач. Система может быть использована на уроках (фрагментарно), на кружковых или факультативных занятиях. Опытная эксплуатация показала, что ученики с удовольствием работают с системой, учителя не испытывают трудностей.

Итоги конкурса

Подведены итоги конкурса на создание учебника «Основы информатики и вычислительной техники».

В результате тайного голосования конкурсной комиссией было принято решение первого места не присваивать. Второе место присуждено рукописи, созданной авторским коллективом в составе **В. Каймина, А. Щеголева, А. Гиглавого, Е. Ерохиной, Д. Федюшина.** Третье место разделили рукописи **А. Верланя, В. Касаткина** и **О. Вершинина.**

Поздравляем победителей!

В следующих номерах мы расскажем о работе конкурсной комиссии и об учебниках, которые были представлены к рассмотрению.

Решено в ближайшем будущем провести новый конкурс на создание учебника информатики.

Л. ШТЕРНБЕРГ

Что ЭВМ может и чего не может понять в вашей программе

В умную вычислительную машину ввели умную программу. Машина ответила «NO ERRORS»*. Тогда юный программист Юра дал ей приказ выполнить программу, ввел исходные данные — и умная машина по умной программе выдала совершенно дурацкий результат. Растерянный Юра пришел к опытному программисту дяде Коле и задал классический вопрос начинающего программиста:

— Дядя Коля, а чего это она?..

— Видимо, есть ошибка, — спокойно ответил дядя Коля, который знал, что наличие ошибки — это нормальное состояние программы.

— Нет у меня ошибок! — возмутился Юра.

— И машина так сказала.

— Ладно, показывай свою программу и рассказывай, как она работает.

Что Юра рассказал	Что было написано в программе
Моя программа решает квадратные уравнения. Здесь *a, b, c* — коэффициенты уравнения, *x*1 и *x* — корни, а вспомогательная переменная *d* нужна для вычисления дискриминанта.	программа квадратное уравнение; действительные *a, b, c, x*1, *x*2, *d*;
Обозначаю корень из дискриминанта через d,	*d*:=квадр_корень (*b***b*—4**a***c*);
ввожу исходные данные и считаю корни.	ввод *a, b, c*; *x*1:=(—*b*+*d*)/(2**a*), *x*2:=(—*b*—*d*)/(2**a*)
Результат печатаю.	вывод *x*1, *x*2;
Вот и все.	конец

— Ну, где здесь ошибка?

— В первом же присваивании: нет в программировании понятия «обозначим», а есть понятие «присвоим», вот и вычисляет ЭВМ значение *d* до ввода значений *a, b* и *c*.

— Хм... А почему машина не говорит, что есть ошибка? Что она, не видит, что значения переменных используются до ввода этих значений? И как она считала результат, если в *a, b* и *c* ничего нет? Что-то же она мне посчитала? Как она складывала и умножала ничего?

И тогда дядя Коля рассказал программистскую сказку.

Нач. Возвращаясь с летней школы юных программистов, Саша заехал на пару дней к бабушке в город, где он раньше не бывал. На следующий день бабушка попросила его съездить в булочную, где продавался самый лучший (на ее взгляд) хлеб, и купить ее любимые булки, а так как города внук не знал, дала ему «программу» поездки: из подъезда пройти направо метров 200, по переулку до трамвайной остановки и 3 остановки на трамвае № 4, чуть подальше и магазин виден. «NO ERRORS» — сказал Саша (что означало «все понял») и поехал.

Через 200 метров направо был... забор. «Так, деление на ноль» — перевел ситуацию на программистский язык Саша. Там был переулок налево, а переулок направо был всего метрах в 100, но куда идти? Пришлось возвращаться. Оказалось, что бабушка ошиблась в оценке расстояния. При попытке выполнить исправленную «программу» сначала возникла ситуация «адресация» (трамвай № 4 там не ходил: выяснилось, что нужен был № 14), затем «переполнение» (через остановку была конечная: бабушка забыла сказать, что надо перейти улицу и сесть на трамвай, едущий в обратном направлении), затем возникло «незаконное преобразование» (магазин около остановки оказался молочным), и наконец, когда «программа» была «отлажена» и Саша попал в нужный магазин, то с первого же взгляда на полки он понял: «конец файла» — любимые бабушкины булки кончились.** Кон.

— Теперь я понял, что «НЕТ ОШИБОК» означает только то, что программа понятна машине, т. е. каждое предписание (оператор) записано правильно само по себе...

— ... и нет внутренних несоответствий типа такого, что предписание цикл есть, а конец цикла — нет, — закончил дядя Коля. — «НЕТ ОШИБОК» означает, что программа синтаксически правильна, а правильна ли она по смыслу — это не проверяется.

— Но неужели трудно проверить, что в моей программе коэффициенты используются до того, как они получили значения?

— А мог Саша из сказки проверить, что через одну остановку у трамвая конечная или что на нужной остановке нет хлебного магазина?

— Ну... если иметь схему транспорта... и чтоб на ней все остановки были указаны... и еще перечень магазинов с адресами... Наверное, все же проще приехать на нужное место и посмотреть.

— Вот именно, — подвел итог дядя Коля.

* «НЕТ ОШИБОК» — один из вариантов сообщения транслятора о синтаксической правильности программы.

** Словом «адресация» обозначается обращение за значением к несуществующему элементу памяти; «перевыполнение» — получение числа (результата); слишком большого для записи в память; «преобразование» — попытка преобразовать в число нецифровые символы; «конец файла» — попытка читать данные с внешнего устройства после того, как все данные уже прочитаны.

— В твоем простейшем случае это еще можно проверить, но если предписание вложено в 2 цикла и 3 ветвления, как установить, что при любом его выполнении все используемые переменные будут иметь значения? И даже если есть случай, когда какая-то переменная окажется без значения, может ли он произойти при реальных исходных данных, на которые рассчитана программа? Поэтому трансляторы не «интересуются» подобными вещами. Кстати, и правильная программа может не дать результатов, если исходные данные неправильны или программа просто на них не рассчитана. Например, твоя не сработает, если корней у уравнения нет, т. е. дискриминант отрицательный; а программа покупки хлеба предполагала, что хлеб есть.

— Ну, с «НЕТ ОШИБОК» все ясно, но как же ЭВМ работала, если в *a, b* и *c* не было никаких значений?

— Сейчас объясню. Только глянь, пожалуйста, на будильник — который час, а то я отсюда не вижу.

— Полшестого. Что??! Ой, я побежал; мама уже, наверное, волнуется!

— Сядь и успокойся. Разве ты не заметил, что будильник стоит: ему «не присвоено никакое значение»?

— Интересно, как я мог это отсюда заметить?

— А как ЭВМ должна заметить, что у какой-то переменной нет значения? Ячейки памяти ЭВМ — не коробки, в которых может что-то быть или не быть. Каждая ячейка напоминает скорее циферблат часов: стоят они или идут, показывают что-нибудь обязательно. Еще точнее, ячейка похожа на ряд лампочек, каждая из которых может либо гореть, либо нет — комбинация зажженных и потушенных лампочек представляет число или символ в закодированной форме, в так называемой двоичной системе счисления, что, впрочем, совершенно неважно. Каждая комбинация что-то означает; все потушенные лампочки — это число ноль.

— Ну вот ноль и означает, что никакого значения нет.

— А чем ноль хуже другого числа? Его тоже можно и складывать и умножать. Кстати, судя по результатам, в твоей программе до ввода у *a, b* и *c* были нулевые значения: то ли ЭВМ «почистила» память перед выполнением, то ли они там остались от предыдущей программы. Интересно, если бы я на будильнике поставил 12 час 00 мин, чтобы показать, что он стоит, ты бы это понял или еще быстрее вскочил?

— Ну, будильник можно просто отвернуть циферблатом к стенке...

— ... т. е. ввести дополнительный признак того, есть на нем показание времени или нет. Это можно, но это дополнительная работа. Именно так сделано в языке Рапира: у каждой переменной есть признак, принимающий значения *пусто* или *число* (или другой указатель типа значения), и прежде чем взять значение, ЭВМ проверяет этот признак. Это требует времени, да и сам признак занимает память. Но Рапира — учебный язык, скорость в ней не так важна, а в производственных языках (Алгол, Фортран, Паскаль, ПЛ/1 и т. д.) такой защиты против ошибок не предусмотрено, чтобы не тратить время и память: ошибся — сам виноват. Ясно?

— Ясно. Дядя Коля, а есть еще ошибки такого сорта?

— Есть, конечно. Вон на школьном дворе выстроились десятиклассники. На правом фланге — X «а» класс, выстроены по списку в журнале. Можешь передать записку моему сыну? В списке он 36-й.

— Конечно, могу. Или здесь опять подвох?

— Подвох. А если я ошибся, и в классе всего 35 человек? К кому попадет записка? К первому из X «б»? Может быть. Но посмотри, между шеренгами X «а» и X «б» промежуток, так что на 36-м месте вообще может никого не оказаться.

— Ясно: в программировании это — выход индекса за границы массива. Но ведь это можно проверить по описанию массива... Впрочем, я, кажется, понял: а я имею доступ к спискам классов в вашей задаче с запиской?

— Безусловно, да: списки классов (т. е. описания массивов) не секреты. А будешь ли ты проверять? Или не станешь тратить время?

— Понял. Если я — язык Рапира, то буду, а если производственный язык, то...

— ... то это зависит от языка. Фортран, который славится скоростью выполнения программ, конечно, проверять не будет; Паскаль может проверить, а может не проверить: при трансляции программы задается режим «включить контроль индексов» или «выключить контроль» — включенный контроль требует сравнительно много времени и памяти; а в ПЛ/1 такое включение — выключение программист пишет прямо в тексте и в одной части программы индексы контролируются, а в другой — нет. Обычно контроль включают при отладке, а потом, когда программа отлажена, выключают, чтобы не тратить на него время.

— А вот еще пример. — Дядя Коля взял распечатку работы программы. — Видишь, написано: «ВРЕМЯ ВЫПОЛНЕНИЯ — 03.51». Сколько работала ЭВМ?

— 3 минуты 51 секунду, — моментально ответил Юра.

— Уверен? А может 3,51 минуты, т. е. 3 минуты и около 30 секунд, а может 3,51 секунды, а может 3 часа 51 минуту?

— Да, действительно. Так сколько же на самом деле?

— На самом деле 3.51 секунды. Именно такие ошибки очень часты при передаче параметров подпрограммам: целые числа, действительные числа, символы, логические значения имеют различные способы кодировки. И если вызывающая программа А передает число, а вызываемая подпрограмма ПР считает, что ей перадали символ, то она расшифрует число как символ, и что при этом получится, сказать трудно. А на твой незаданный вопрос «Неужели ЭВМ не видит несоответствия?» сразу отвечу: не видит потому, что программа А вводится и транслируется сегодня, а подпрограмма ПР, может быть, будет вводиться завтра или введена вчера. И даже если они введены вместе, откуда ЭВМ знать, что программа А будет вызывать именно эту подпрограмму ПР? Может, эта подпрограмма ПР написана твоим другом и будет вызываться его программой?

— Ну а почему нельзя вместе с фактическим параметром передавать и его тип? Что стоит написать не «3.51», а 3 мин 51 сек»?

— Что стоит? В программировании на все одна цена: время и память. Рапира, Паскаль, Алгол-60 передают вместе с фактическим параметром его паспорт, в котором написано о его значении все, что надо, а подпрограмма проверяет этот паспорт и, если значение ей не подходит, выдает предупреждение. Фортран и ПЛ/1 берут значение не глядя и расшифровывают его по описаниям, имеющимся в подпрограмме, так же, как ты по своим представлениям решил, что «3.51» — это 3 минуты 51 секунда, не спросив, а какие представления на этот счет у ЭВМ.

— Я читал, что в Фортране при несоответствии типов формальных и фактических параметров возникает ошибка. Но нигде не написано, какая.

— Во всех случаях, которые мы сегодня разбирали, возникает ошибка, для которой программисты применяют термин *эффект не определен*. Иначе говоря, ЭВМ эту ошибку не замечает и продолжает работу; в описании языка не сказано, что надо делать в этом случае, поэтому разные ЭВМ имеют полное право реагировать на нее по-разному, более того, одна и та же ЭВМ при одних и тех же исходных данных может давать разные результаты. Например, в твоей программе: ячейки памяти, отведенные под *a, b* и *c*, могут хранить значения от той программы, которая выполнялась на ЭВМ раньше. Понятно, что нет гарантии, что вчера, сегодня и завтра останутся одни и те же значения.

— Ну а как все-таки бороться с такими ошибками?

— Аккуратностью и умением вовремя задать себе вопрос: «Я это знаю, но откуда это знает ЭВМ?»

О. ТУГОЛБАЕВ, П. ПАНКОВ, И. ДОЛГОПОЛОВА
г. Фрунзе

Олимпиады по основам информатики и вычислительной техники в Киргизии

С 1985 г. в столице Киргизии проводятся городские олимпиады по ОИВТ. В этом году Министерством просвещения совместно с Академией наук и Министерством высшего и среднего специального образования проводилась Первая Республиканская олимпиада школьников по ОИВТ.

Она проходила в два этапа. В ходе первого (отборочного) по результатам учебной, кружковой и факультативной работы определялись участники второго — собственно олимпиады.

Для школьников столицы отборочным этапом служила городская олимпиада, в подготовке и проведении которой традиционно приняли участие Институт математики АН республики, Киргизский государственный университет и Фрунзенский политехнический институт.

3-я городская олимпиада столицы также состояла из двух этапов: на первом участникам районных математических олимпиад было предложено несложное задание (см. задачу № 4), по результатам выполнения которого члены жюри производили отбор учащихся для участия в городской олимпиаде. Отдельно отбор был произведен в специализированной школе № 61 им. И. В. Курчатова.

Число участников городской олимпиады, набранных таким образом из IX—X классов, в течение трех лет колебалось около 90. На олимпиаду допускались и некоторые учащиеся VII—VIII классов из числа детей, проявивших себя в самостоятельном изучении нового предмета.

Задачи составлялись на основе разработанных ранее сотрудниками ИМ АН республики совместно с КиргизНИИ педагогики методических правил и подбирались таким образом, чтобы при их решении учащийся смог пройти все основные этапы от постановки задачи до отладки программы. Для выполнения заданий было достаточно иметь знания, предусмотренные школьной программой по ОИВТ, при условии ее творческого усвоения.

Республиканская олимпиада включала два тура: теоретический и практический (в компьютерном классе). Для участия в ней учащиеся отбирались по двум группам: первая — «начинающие» (первый год обучения и самостоятельно изучающие ОИВТ), вторая — изучающие предмет более года; в порядке исключения допускались ученики VII—VIII классов. Всего участвовало свыше 100 человек.

На теоретическом туре было предложено четыре задачи (4 астрономических часа). На практическом туре (1,5 часа) — два задания для решения на ПЭВМ («Агат» и «Ямаха») на языке Бейсик; участникам разрешалось запрашивать консультации у членов жюри (только по техническим вопросам, а не по способу решения).

Для уравнивания шансов учащихся городских и сельских школ в первый день олимпиады (после теоретического тура) каждому участнику предоставили возможность освоить работу на ПЭВМ под руководством опытных преподавателей (кстати, многие участники олимпиады из сельских школ обучались в лагерях по освоению ЭВТ летом 1986 г.).

Анализ результатов показал следующее:

Участники олимпиады умеют решать нестандартные задачи (например, № 6). Некоторые из них, как и предполагалось, выучили элементы

Бейсика в ходе выполнения заданий (типа № 5). Многие писали решения задач I тура на Фортране и Бейсике, но еще нетвердо знают правила и конструкции школьного языка информатики. Даже лучшие не смогли коротко и ясно изложить идею алгоритма (перед его подробным описанием на одном из алгоритмических языков).

Приведем (со сквозной нумерацией) условия некоторых задач.

Городская олимпиада, 1985 г.

1. Заданы натуральные числа M1, M2, M3. Найти минимальное натуральное число, не представимое суммой этих чисел. Сумма может состоять из одного слагаемого; каждое из чисел может входить в нее только один раз.

Районные олимпиады, 1986 г.

2. Даны числа C, H, P, T, обозначающие длины четырех стержней. Можно ли составить из этих стержней параллелограмм?

Городская олимпиада, 1986 г.

3. Написать на любом алгоритмическом языке программу, которая находит размеры прямоугольного ящика (без крышки) с наименьшей площадью поверхности, в который можно поместить 1986 одинаковых кубиков так, что их грани будут параллельны граням ящика.

Районные олимпиады, 1987 г.

4. По телефону передаются две (десятичные) цифры и для контроля — последняя цифра их суммы и разность между большей и меньшей цифрами. Предполагается, что неправильно понята может быть самое большое одна цифра из этих четырех. Написать (на любом алгоритмическом языке) алгоритм, который:

а) определяет, была ли какая-нибудь цифра неправильно понята;

б) исправляет допущенную ошибку.

Городская олимпиада, 1987 г.

5. а) Что делает данная программа, написанная на языке Бейсик? б) Можно ли той же цели добиться более простым путем?

```
 5 INPUT A, B, C
11 IF A>B THEN 35
15 IF A=B THEN 21
20 IF C>A THEN 30
21 PRINT 'НЕТ':STOP
30 IF C<B THEN PRINT C:STOP
32 PRINT 'НЕТ':STOP
35 D=A:A=B:B=D:GO TO 20
```

Республиканская олимпиада, 1987 г.

6. Киргизский алфавит отличается от русского добавлением букв Ң (носовое Н), Ө и Ү (смягченные О, У). Вопросительное наклонение в киргизском языке образуется добавлением к слову двухбуквенного окончания: мугалимби, советпи, жээнби, нанбы, биринчиби, көппү, койбу, акпы, тообу, көчөбү, суубу, сүтпү, бетпи, канатпы, бутпу, асылбы, болотпу, ысыкпы, эртенби, кыялбы, коёнбу, парашютпу.

Написать алгоритм, составляющий по заданному слову соответствующее окончание.

7. Написать алгоритм, который находит все возможные расстановки пяти белых и трех черных кружков в ряд.

8. Будем говорить, что число X близко к Y, если $|X-Y|<0{,}1987$. Задан массив (таблица) B из K вещественных чисел.
Написать алгоритм, который вычеркивает из массива некоторые числа так, что все оставленные далеки одно от другого, а каждое из вычеркнутых близко к одному из оставленных.

9. Клетки шахматной доски занумерованы от 1 до 64 по строчкам слева направо и снизу наверх. Написать и отладить программу, которая по заданному номеру клетки дает номера всех клеток, имеющих с ней общую сторону.

ЧТО МОЖЕТ ЭВМ?

«Вроде что-то припоминаю...»

«Вот и у нас был похожий случай!» — такие или похожие слова нередко звучат в разговорах. Способность к установлению ассоциаций — один из мощных инструментов мозга. Она же, пусть и в примитивной форме, полезна и для ЭВМ.

Обычные элементы оперативной памяти — запоминающие устройства с произвольной выборкой (ЗУПВ) — работают «по адресу»: на вход ЗУПВ подается закодированный адрес ячейки памяти, на выходе появляется число — содержимое этой ячейки. Представьте, что в ОЗУ записаны разные «случаи», объединенные в группы «похожих». Каждая группа имеет свой номер, который ставится в начале описания каждого «случая». Чтобы найти все «случаи», входящие, например, в группу 5, нам придется просмотреть все ячейки памяти: в каких из них записана пятерка?

Но кроме ЗУПВ существуют еще и ассоциативные запоминающие устройства (АЗУ), работающие «по данным». На их вход подается число, а на выходе появляются адреса ячеек, в которых оно записано (если записано). В такой памяти найти все пятерки можно очень быстро.

А вот и практическое применение АЗУ: устройство управления данными Am95C85 компании «Эдвансд микро дивайсез» (США) позволяет обрабатывать файлы в 500 раз быстрее, чем программно реализованные алгоритмы и ЗУПВ.

Гипотезы

В. СЕРГИН
Доктор физ.-мат. наук

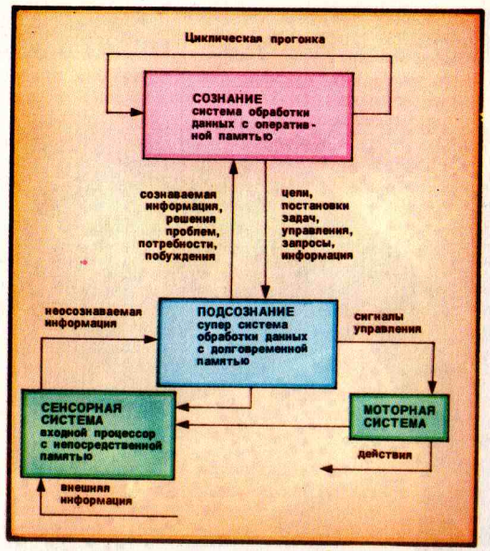
Механизм формирования высших функций мозга, таких как сознание, творчество и мышление в целом, представляют собой одну из фундаментальных тайн природы, которая давно привлекает внимание специалистов различных областей знания. В настоящее время широко распространено представление о том, что наш мозг функционирует подобно большой ЭВМ. Несмотря на то, что такое представление аргументировано, оно встречает обоснованные возражения, связанные с существованием некоторых важных свойств человеческого мышления, которые не находят удовлетворительного объяснения в рамках модели мозга как вычислительной системы. К ним относятся интуиция, подсознательные побуждения, управляющие поведением, эмоциональные оценки сложных явлений (такие, как чувство гармонии или красоты) и другие еще более таинственные свойства мозга, которые, возможно, простираются вплоть до гипноза и так называемого сверхчувствительного восприятия.

Все это побуждает многих специалистов рассматривать ЭВМ лишь в качестве модели простых (рутинных) свойств мозга. При этом наиболее глубокие и важные свойства человеческого мышления остаются необъясненными, что сохраняет почву для околонаучных спекуляций и всякого рода фантастических измышлений. Поэтому рациональное объяснение пока еще загадочных феноменов мышления и психики принципиально важно для науки вообще и для понимания природы человеческого интеллекта в частности.

Блок-схема модели интеллекта

Наши рассуждения начнем с, казалось бы, мало продуктивных представлений о том, что мозг — это машина мышления, которая перерабатывает информацию и функционирует подобно большой вычислительной системе. Это не означает предположения, что человеческий мозг устроен и работает как существующие ЭВМ. Просто вычислительная система представляет собой наиболее совершенную машину, предназначенную для обработки информации и в этом смысле является ближайшей аналогией машины мышления.

В качестве основы для дальнейших рассуждений введем предположения о макроструктуре интеллекта:



интеллект состоит из двух связанных, но функционально различных систем обработки данных (вычислительных систем) — сознания и подсознания;

объем памяти и производительность подсознания как вычислительной системы на много порядков больше, чем аналогичные характеристики сознания.

Предположение о том, что сознание функционально обособлено от подсознания и характеризуется определенными (необязательно фиксированными) границами локализации, имеет нейрофизиологические основания. Считается, что основным априорным условием сознательного состояния является способность обозревать свое внутреннее состояние посредством периодической передачи на вход имеющейся в памяти информации [1]. Повторный ввод хранящейся в памяти информации позволяет распознавать ее и сравнивать с содержимым памяти. Тем самым осуществляется просмотр формируемых внутри образов (моделей) в потоке внешней информации. Следовательно, процесс осознания представляет собой ассоциативное воспоминание с обновлением и требует периодического распознавания информации, представляющей внутреннее состояние (образ) и внешнюю среду (реальный мир).

В настоящее время неизвестно, на каких структурах мозга реализуется сознание. Тем не менее, можно утверждать, что области мозга, охваченные циклическими процедурами распознавания и отождествления собственного «я» и образа внешнего мира, представляют собой место локализации сознания. Циклическая прогонка хранящейся в памяти информации должна осуществляться с частотой, превышающей характерные скорости изменения внешней среды, что необходимо для слитного восприятия изменений. Столь частая прогонка содержимого памяти накладывает жесткие ограничения на ее объем. Кроме того, ресурсы сознания как вычислительной системы используются здесь не столько для выполнения полезной работы — обработки потока внешних сигналов и выработки поведенческих реакций, сколько для периодического распознавания и обновления модели собственного «я» и модели внешнего мира. Поэтому производительность сознания как системы обработки внешней информации и принятия решения должна быть небольшой.

На рисунке представлена блок-схема модели интеллекта, построенная в соответствии с высказанными предположениями. Ее компоненты и связи между ними можно охарактеризовать на основе данных психологии и нейрофизиологии.

В непосредственной памяти содержится почти вся информация, поступающая на органы чувств, но хранится она всего несколько сотен миллисекунд. При переводе информации из непосредственной памяти в оперативную (память сознания) происходит ее селекция в соответствии с содержанием решаемой задачи. Объем оперативной памяти (количество запоминаемых символов) ограничен магическим числом Миллера 7 ± 2. Время хранения информации составляет в среднем несколько десятков секунд. Потеря информации происходит по мере того, как она перестает активно использоваться. Повторение является эффективным способом удержания информации в оперативной памяти и важным условием ее перевода в долговременную память.

Емкость долговременной памяти, по-видимому, обеспечивает хранение всей информации, потенциально полезной для наилучшей адаптации вида к изменяющимся условиям окружающей среды. В долговременной памяти содержатся формы поведения, заложенные генетически и приобретенные в индивидуальном развитии; модели объектов и пространство образов, сформированные в процессе обучения; процедуры обнаружения и идентификации объектов; правила использования хранимой информации и многое другое, необходимое для организации поведения [2]. В долговременной памяти подсознания содержатся глубоко усвоенные этические и социальные нормы, жизненный опыт, доведенные до автоматизма профессиональные и бытовые навыки [3] и другие знания (процедурные и декларативные), выражающие личность и индивидуальность человека.

Время хранения информации в долговременной памяти, по-видимому, соответствует продолжительности жизни. Возможно поэтому нейроны, в отличие от большинства других клеток организма, никогда не обновляются, что обеспечивает сохранение информации и вычислительной структуры мозга на протяжении длительного времени без искажений. Долговременная память имеет, по-видимому, иерархическую организацию с различными временами выборки информации. В типичных случаях выборка информации осуществляется вне прямого контроля сознания в соответствии с целью и содержанием конкретной деятельности.

Органы восприятия несут в мозг большой поток информации. Сигналы отличаются по амплитуде, фазовым и частотным характеристикам, комбинациям поступления от различных органов, смысловой нагрузке и другим признакам. Ясно, что сознание должно быть защищено от избыточной информации, поток которой должен соответствовать его ограниченной производительности. Следовательно, в область сознания должна пропускаться лишь функционально необходимая и предварительно обработанная (селектированная, отфильтрованная, агрегированная) информация. Поток информации от органов чувств в подсознание не требует жестких ограничений, и его величина может быть на много порядков больше осознаваемого потока. Это значит, что внешний мир, постигаемый подсознанием, может быть гораздо более богатым, разнообразным и тонко структурированным, чем его упрощенное отражение в сознании.

Мотивы, побудительные импульсы, целевые установки и другие управляющие решения, вырабатываемые подсознанием, приходят в область сознания без какой-либо аргументации и трактуются как эмоциональные. На этом и основаны существующие в психологии представления об эмоциональном управлении мыслью. Однако, по нашей концепции, решение проблем и выработка управляющих стимулов осуществляются в подсознании на основе строгих процедур и являются результатом обработки громадного объема информации.

Рассматриваемая модель интеллекта вполне согласуется с нейробиологическими данными о структуре и клеточных механизмах головного мозга. По данным Маунткасла [1], новая кора переднего мозга содержит 50 млрд. нейронов, организованных в 600 млн. вертикальных миниколонок, функционирующих параллельно. Очевидно, что производительность матричного процессора столь высокой размерности должна быть чудовищной. По современным данным [1], значительная часть новой коры используется для параллельной обработки различных видов сенсорной и моторной информации. Поскольку функции восприятия связаны с универсальными процедурами распознавания и запоминания информации, эти же матричные структуры, в принципе, могут использоваться и при выполнении других высших функций мозга. Вероятно, матричные структуры могут, в различные интервалы времени, использоваться для выполнения различных функций. Не подтверждает ли это предположение тот факт, что когда человек глубоко задумается, он не видит и не слышит, что делается вокруг. По-видимому, в это время соответствующие области коры лишь отчасти выполняют свои функции по обработке внешней информации, а их более глубокие слои участвуют в интенсивных процессах мышления. Каждому, вероятно, знакомо побуждение отключаться от внешней информации, чтобы поразмышлять в свое удовольствие.

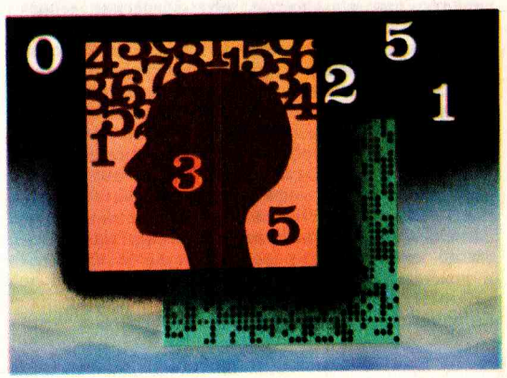
Структура творческого процесса

Итак, в соответствии с высказанными представлениями, модель интеллекта представляет собой две взаимодействующие системы обработки данных. Одна из них (сознание) имеет сравнительно небольшую память и производительность, что ограничивает размерность задач, которые она способна решать. Объем памяти и производительность другой (подсознания) позволяет решать задачи высокой размерности. Если бы это было в самом деле так, то решения, полученные в подсознании, могли бы доставлять сознанию принципиально новые факты, которые сознание самостоятельно получить не может.

Невысокая производительность сознания как вычислительной системы хорошо известна из повседневного опыта. В уме сравнительно успешно решаются задачи, включающие не более 2—3 влияющих факторов. Лучше всего нам удаются однофакторные задачи. Если необходимо быстро принять решение в условиях, когда одновременно возникает большое количество различных обстоятельств, мы можем почувствовать, что «голова пошла кругом». Это означает функциональный отказ сознания как вычислительной системы. С математической точки зрения разумным выходом в этой ситуации может быть понижение размерности задачи. Обычно так и поступает человек, который не теряется в трудных ситуациях. Он последовательно перебирает в уме и оценивает различные обстоятельства, отбрасывает второстепенные, а затем принимает решение, руководствуясь главными факторами. Такой прием позволяет защитить рассудок от переполнения информацией, но, разумеется, не гарантирует достижения наилучшего решения.

Объем памяти и производительность подсознания много больше, что позволяет решать задачи высокой размерности, учитывая большое число влияющих факторов. В таком случае, в области сознания может формулироваться лишь постановка задачи. Процедура исследования осуществляется в подсознании и затем готовое решение выдается в область сознания. В этом, по-видимому, и состоит природа интуиции (озарений, внезапных открытий, догадок). В область сознания может выдаваться лишь окончательное решение задачи, которое воспринимается сознанием как «интуитивное». Промежуточные операции отыскания решения не могут выводиться в сознание из-за его малой информационной емкости.

Интуиция характеризуется тем, что позволяет устанавливать новые связи и отношения в условиях видимого недостатка данных. В работе [3] указывается, что в генезисе различных видов интуиции (интуиции — догадке, интуиции — прямом усмотрении истины) есть нечто принципиально общее, а именно дефицит сознаваемой информации, необходимой и достаточной для логически безупречного заключения. Другие важные свойства интуиции состоят в том, что она непредсказуема и в то же время неслучайна. Из предложенной выше схемы формирования интуитивного решения естественным образом вытекают перечисленные свойства интуиции. В самом деле, решения, полученные путем логически безупречных заключений, являются строгими результатами



работы сознания. Строгие результаты работы подсознания, как системы обработки данных, обладающей громадным неосознаваемым объемом информации и высокой производительностью, позволяют устанавливать новые связи и отношения, непредсказуемые для сознания. Интуитивное решение непредсказуемо для сознания в том же смысле, в каком решение супер-ЭВМ непредсказуемо для мини-ЭВМ. В то же время интуитивное решение неслучайно, оно определяется объемом и специфичностью накопленных индивидуумом знаний, вычислительной структурой его интеллекта и сознательно сформулированной постановкой задачи (целью, вопросами).

Предложенная модель интеллекта вполне соответствует представлениям о творческом процессе, сложившемся в современной когнитивной психологии. Определение цели исследования, сбор данных и формулировка задачи являются сознательными процессами. Затем следует интуитивное решение проблемы. И лишь после того, как ответ (не обязательно правильный) стал известен, производится его аргументированное обоснование (или опровержение). Существенной стороной процесса мышления является то, что принципиально новые факты устанавливаются вне сферы сознательной деятельности, в подсознании. Модели окружающего мира строятся здесь на основе недоступной сознанию информации, имеют много более высокую размерность и могут описывать весьма тонкую структуру объектов и их взаимоотношений. Установленные с их помощью закономерности могут быть одновременно неожиданными для сознания и правильными по существу. Процедура доказательства интуитивного решения или его опровержения путем логического анализа или экспериментальной проверки являются функциями сознания.

Из того обстоятельства, что подсознание оперирует множеством факторов и информацией, недоступных сознанию, а промежуточные выкладки не могут выводиться в область сознания, следует, что строгое доказательство или аргументированное обоснование интуитивного решения требует специальной работы сознания. В некоторых случаях это должна быть громадная и нетривиальная работа, в процессе которой также необходимы творческие озарения. Поскольку такая работа не всегда завершается успешно, в науке неизбежно должны существовать представления (или утверждения), истинность которых подтверждается всеми известными фактами, но не имеет строгих доказательств. Примерами могут служить знаменитые математические теоремы, часть которых получила доказательства лишь через несколько десятилетий или столетие после публикации. Иногда требуется построение новых разделов математики, чтобы ранее несвязанные или неуловимые для сознания факты стали компонентами стройных теорий, и только тогда доказательство давно сформулированных теорем становится доступным сознанию. Множество подобных примеров дают физические теории. Открытия Г. Галилеем относительности движения и независимости ускорения свободного падения от массы не имели строгих доказательств, а его идеи и доводы отвергались и оспаривались вплоть до работ И. Ньютона. Уравнения электродинамики были выписаны в окончательном виде без доказательства их адекватности реальным физическим процессам, а рассуждения Дж. Максвелла, будто бы поясняющие их вывод, никто всерьез не принимал. И такое перечисление можно продолжить.

Следует заметить, что описанная структура творческого процесса является лишь грубой схемой. В реальном творчестве взаимодействие между сознанием и подсознанием протекает непрерывно, проявляясь в более тонкой структуре процесса исследования. Поэтому выделенные этапы рассудочной и интуитивной работы мозга весьма условны. Так, например, в науке широко распространено представление о том, что правильно поставить задачу, значит наполовину решить ее. Постановка задачи, будучи преимущественно сознательным процессом, содержит важные эвристические компоненты, что свидетельствует об участии подсознания в этой работе. В правильно поставленной задаче содержится «предчувствие» характера ее решения, что свидетельствует об уже выполненной (в подсознании) предварительной работе. С другой стороны, в решении проблемы, даже если оно имеет интуитивный характер, сознание может играть важную координирующую роль. Оценка и проверка истинности полученных результатов, в свою очередь, не являются чисто сознательными процессами. Например, при экспериментальной проверке суждение о достаточности экспериментальных данных является интуитивным. При оценке истинности теории иногда ссылаются на эстетические критерии, такие как чувство красоты. (Известно высказывание Р. Фейнмана: «Истину можно узнать по ее красоте»). Характерной особенностью критерия красоты является осознание совершенства как некоторого факта и, одновременно, невозможность четко сформулировать причины такого суждения. С точки зрения развиваемой концепции такой критерий истинности научной теории как чувство красоты представляет собой многоцелевой показатель, формируемый в подсознании.

Объяснение некоторых феноменов психической деятельности

Состояние сознания достигается дорогой ценой, за счет потери производительности, а природа обычно оптимизирует свои функции. Поэтому естественно предположить, что сознание постоянно реализуется лишь в сравнительно небольших областях головного мозга, которые расширяются только при интенсивной рассудочной деятельности. В зонах, которые в данный момент граничат с областями сознания, имеет место краевое сознание, а в остальных частях мозга идет более производительная подсознательная работа. Расширение области сознания может происходить только за счет подсознательной сферы.

Область сознания сокращается до минимума во время сна (чем, по крайней мере, отчасти, определяется его физиологическая необходимость) и мозг освобождается для более производительной работы. Это может быть внутренняя работа, которая необходима в любой системе обработки информации, например, упорядочивание массивов данных, доработка и перестройка моделей и др. Во сне могут решаться проблемы, которые были поставлены ранее в процессе взаимодействия сознания и подсознания. По данным нейрофизиологии, интенсивность работы мозга в целом постоянна днем и ночью. Но во время сна снижаются до минимума обработка внешней сенсорной информации, управление моторикой и процедуры осознания. Следовательно, во время сна интенсивность работы мозга как автономной вычислительной системы возрастает. Это подтверждается хорошо известными фактами, когда решение трудных проблем приходит в утреннюю пору без видимых усилий, как бы самопроизвольно: подсознание выдает результаты ночной работы. Пословица «утро вечера мудренее» отражает богатейший экспериментальный материал, накопленный в процессе разносторонней человеческой деятельности.

Состояние, при котором область сознания может сильно сокращаться, освобождая мозг для более продуктивной работы, бывает и во время беззаботного отдыха. В литературе описаны многочисленные случаи появления идей и спонтанного решения ключевых научных проблем во время отдыха, прогулки в лесу, в горах, и тому подобных обстоятельствах, когда их авторы, казалось бы, вообще не думали о работе.

Из относительной обособленности двух систем обработки данных, сознания и подсознания, следует возможность двойственности в мироощущении и поведении человека. В случае больших отклонений от нормы такая двойственность может приобретать черты феномена, известного в классической психологии как раздвоение личности (патологические состояния расщепления личности могут иметь несколько более сложную природу). Мы ограничимся примером нормального состояния, когда хорошо выраженная двойственность является естественным следствием относительной обособленности сознания и подсознания, и воспользуемся с этой целью фрагментом из рассказа А. И. Куприна «Лунной ночью».

«Случалось вам видеть во сне, будто вы сдаете трудный экзамен? Вам задают вопрос, и вы на него никак не можете ответить. Вы усиленно думаете, ломаете голову, но ответ, как нарочно, не идет на ум. Тогда учитель обращается к одному из ваших товарищей, тот отвечает самым правильным и блестящим образом, и вам становится стыдно за ваше незнание. Случалось это с вами?

...Во сне на экзамене отвечает ваш товарищ. Но ведь товарища-то на самом деле нет, отвечаете вы же и вы же удивляетесь тому, что говорите. Видите какая двойственность?»

Здесь процедура экзамена и собственное поведение моделируются в сознании (как это бывает и в бодрствующем состоянии). Усилие вспомнить забытый материал и вопросы обращены к подсознанию. Ответ относительно обособленной системы, подсознания, является внешней для сознания информацией и трактуется, в соответствии со сценарием экзамена, как ответ товарища. В бодрствующем состоянии ответы подсознания также иногда трактуются как подсказки со стороны или внушения свыше. Так, например, Сократ был уверен, что его идеи были ему продиктованы его личным демоном, а Помпилий часто «консультировался» у нимфы по имени Эгерия [4]. Таинственные голоса Жанны д'Арк и многих других подвижников имеют вполне материальную природу и их информационным источником является система обработки данных, именуемая подсознанием.

Поток информации от органов чувств в подсознание на много порядков превышает поток информации в область сознания. Их разница дает величину неосознаваемой информации, которая никак не регистрируется сознанием. Так называемое «внечувственное восприятие» отчасти объясняется в рамках рассматриваемой модели как часть потока неосознаваемой информации. Заметим, что рассматриваемой концепции не противоречит и возможность существования пока еще неизвестных науке органов восприятия, сигналы которых используются только в подсознательной сфере.

Явление гипноза можно объяснить в рамках рассматриваемой модели, если предположить существование способов воздействия, расстраивающих специфические процедуры осознания. Тогда область сознания может искусственно сокращаться до минимума или блокироваться внешними воздействиями. В этих условиях указания гипнотизера могут восприниматься непосредственно подсознанием гипнотизируемого как программа, подобная той, что обычно поступает от собственного сознания и потому выполняется. При гипнотическом внушении формируется интеллект-кентавр, состоящий из сознания гипнотизера и подсознания гипнотизируемого. Гипнотизер, не вызывая нарушений в основных свойствах функций мозга гипнотизируемого, получает доступ к внутренним структурам, причем все ассоциативные связи и все хранилища информации о прошлом опыте остаются незатронутыми, а контакт этих структур с внешним миром ограничивается только одним путем, который выбирает гипнотизер [5]. Т. е. сознание гипнотизера служит интерфейсом между подсознанием гипнотизируемого и внешним миром. Но хотя сознание гипнотизируемого подавлено (блокировано), его личность и индивидуальность сохраняются (поскольку носителем этих фундаментальных свойств является подсознание). Заставить гипнотизируемого совершить под влиянием внушения поступки, которые согласно его нормам поведения или представлениям о безопасности являются непристойными или опасными, удается лишь в очень небольшом числе случаев [5].

Заключение

Главным результатом представленной здесь работы является то, что чисто человеческие черты творческого мышления и весьма загадочные феномены психической деятельности удается объяснить с точки зрения функционирования мозга как системы обработки данных (вычислительной системы). При этом не делается специальных допущений для каждого данного случая. Чрезвычайно широкий спектр феноменов мышления и психики объясняется с единых позиций, на основе предложенной модели интеллекта. Причем введенная макроструктура интеллекта не содержит каких-либо экзотических или рискованных допущений, элементарно проста и основана на хорошо известных эмпирических фактах психологии и нейрофизиологии. Однако несмотря на свою простоту, предложенная модель позволяет по-новому взглянуть на природу человеческого мышления и высказать некоторые новые предполо-

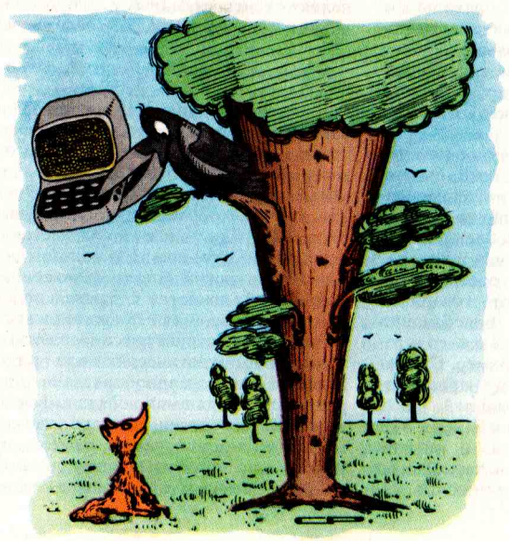
жения о функционировании мозга, которые могут быть проверены экспериментально.

Система «сознание — подсознание» представляет собой оптимальную структуру, отработанную процессом эволюции. С одной стороны, сознание не перегружено информацией, что позволяет приближенно, но в темпе реального времени моделировать свое поведение в окружающем мире. С другой стороны, используется вся доступная информация, которая обрабатывается в подсознании и откуда готовые результаты передаются в область сознания. Представляется, что дальнейшая разработка этой схемы и детализации модели интеллекта могут привести к новым идеям в области построения эффективных систем обработки данных.

Литература

1. *Эделмен Дж., Маунткаса В.* Разумный мозг. М., 1981. 133 с.
2. *Дудкин К. Н.* Зрительное восприятие и память. М., 1985. 206 с.
3. *Симонов П. В.* Мотивированный мозг. М., 1987. 237 с.
4. *Адамар Ж.* Исследование психологии процесса изобретения в области математики. М., 1970. 150 с.
5. *Уолтер Г.* Живой мозг. М., 1966. 300 с.

Веселый урок



Точка зрения

А. МАТЮШКИН-ГЕРКЕ

Каким быть школьному курсу информатики

Исключительная важность проблемы компьютеризации образования заставляет с пристальным вниманием следить и за конкретными шагами, направленными на ее решение, и — особенно! — за решением вопросов стратегического плана, связанных с выбором путей продвижения к цели.

В этой статье речь идет о курсе ОИВТ, призванном сейчас играть центральную роль в решении поставленной перед средней школой задачи: «Привить учащимся навыки пользования компьютерами и вооружить их знаниями о широком применении этой техники в народном хозяйстве» (Правда. 1985. 2 февраля).

К настоящему времени Минпросом СССР опубликованы две программы этого курса: так называемые ныне действующая (Математика в школе. 1985. № 3) и конкурсная (Математика в школе. 1986. № 3; Информатика и образование. 1986. № 1). Первая из них получила солидное методическое обеспечение: пробное учебное пособие в двух частях, методическое пособие для учителей в двух частях, ряд ориентированных на нее методических статей и рекомендаций в упомянутых выше журналах, поурочные разработки в «Учительской газете» и т. д. Идейно родственная ей конкурсная программа в основном отличается ориентацией на систематическое использование в учебном процессе некоторой модели школьной ЭВМ.

К сожалению, обе программы, разработка которых велась без сколько-нибудь достаточного участия широкой научно-педагогической общественности, оказались совершенно неудовлетворительными.

Основной их порок — неминуемая полная дезориентация учащихся в узловых вопросах, связанных с практическим применением ВТ. *Настойчиво проводимая в этих моделях курса линия на предъявление обучаемым «умений алгоритмизации» в качестве якобы необходимого и достаточного условия успешности взаимодействия человека с ВТ полностью ошибочна.*

На самом деле, во-первых, совершенно неопределенным является само понятие «умений алгоритмизации». Даже если ограничиться задачами с идеально полной и четкой постановкой, то и среди них, наряду с простейшими, найдутся такие, что алгоритмизировать их решения не сможет даже специалист высокого уровня. Для разных предметных областей и «умения алгоритмизации» потребуются совершенно разные. Говорить об «умении алгоритмизации» безотносительно к конкретным предметным областям и уровню сложности задач просто нелепо!

Во-вторых, полнота и четкость постановки прикладных задач возникают лишь в результате целого ряда допущений и упрощений, абстрагирования от тех или иных факторов, которые от этого отнюдь не перестают действовать. Насколько далеко от реальной действительности могут отстоять алгоритмические построения, даже выполненные на очень высоком уровне, можно видеть хотя бы по публикациям акад. Л. В. Канторовича и Н. Н. Моисеева, в которых они рассказывают о неудачных попытках автоматизировать управление транспортными системами. Должной отдачи от многочисленных АСУ мы тоже не имеем вовсе не потому, что плохи там программы и алгоритмы! Можно ли в подобных случаях считать «успешным» взаимодействие человека с вычислительной техникой на том основании, что, дескать, алгоритмы составлены грамотно и полностью соответствуют постановке задач?!

В-третьих, применение ВТ как одного из (!) средств решения прикладных задач, не начинается и не заканчивается алгоритмизацией. Гораздо более сложными и трудоемкими оказываются другие этапы: постановка задачи, выбор методов решения (с учетом имеющихся ресурсов), его информационное и организационное обеспечение. Более того, те задачи, которые действительно успешно решаются на основе полной алгоритмизации, совсем не самые сложные. Очень многое приходится делать на далеко не полностью формализованной основе. Могущество человеческого интеллекта проявляется здесь не столько в умении алгоритмизировать те или иные компоненты процесса решения, сколько в способности по ходу этого процесса оценивать ситуацию и осуществлять выбор путей его продолжения.

Подчеркнем, что такой выбор производится, как правило, отнюдь не на основе заранее заданных критериев и не из заранее очерченного множества вариантов, так что возможности алгоритмизации оказываются здесь принципиально ограниченными! Разумеется, с течением времени мы сможем усовершенствовать подходы к решению тех или иных задач, но если впоследствии благодаря этому «уровень алгоритмизации» удастся повысить, то к тому времени уже будут настоятельно требовать своего решения новые, еще более сложные задачи. И полная алгоритмизация опять-таки окажется недостижимой.

Школьнику, воспитанному, образно говоря, в духе преклонения перед алгоритмизацией и нигилизма по отношению ко всем остальным аспектам применения ЭВМ, успешно взаимодействовать с вычислительной техникой будет куда труднее, чем даже человеку, совсем не изучавшему информатики. Ведь, прежде чем понять истинные закономерности такого взаимодействия, такому школьнику надо будет освободиться от груза ошибочных представлений, что дается не просто и не безболезненно.

Принципиально ошибочным является и использование так называемого «алгоритмического языка» в качестве стержневой основы определяемого этими программами курса. Прежде всего, недопустимо предъявлять учащимся предлагаемый его авторами способ описания алгоритмов как единственно возможный, а о существовании других даже не упоминать. Иначе, как грубой дезинформацией это назвать нельзя. Никакой «всеобщностью», «универсальностью» этот язык на самом деле не обладает. Так, он абсолютно неприспособлен для описания взаимодействия между пользователем и исполнителем по ходу самого процесса решения, что важно даже на школьном уровне обучения. Отсутствуют в нем и многие другие возможности, необходимые для описания алгоритмов решения различных классов решаемых на практике задач. Поэтому представлять его в качестве единственного, обязательного к использованию средства неправомерно.

Направленность этого языка на привитие учащимся отдельных начальных навыков структурного программирования (что совсем не актуально для общеобразовательного курса) ни в коей мере не компенсирует такой его существенный недостаток, как трудность усвоения при первоначальном знакомстве с предметом.

Противоречат друг другу описание понятия алгоритма, приводимое в пробном учебном пособии, и характеристики, приписываемые авторами этому «алгоритмическому языку». Стоя на позициях указанного описания, придется признать, что этот язык служит отнюдь не для записи самих алгоритмов (как это утверждают авторы), а всего лишь для более или менее детализированного описания их структуры. Ведь если исполнителем является ЭВМ, то и под алгоритмом мы тогда должны были бы понимать не что иное, как ориентированную на эту ЭВМ программу. Нелепо пытаться уйти от признания этого факта, вводя дополнительные понятия типа «человек-исполнитель», и не только потому, что (если не делать дополнительных оговорок) под человеком с равным основанием можно понимать и профессионала высокого класса, и малолетнего ребенка, и неграмотную старушку. Главное — неправомерно игнорировать направленность приводимых построений на использование в конечном счете именно средств ВТ.

Разделы этих программ, посвященные современным программным средствам (ПС), их сущности, проблемам создания и использования, к сожалению, тоже неудовлетворительны. Прежде всего, нельзя согласиться с тем, что те из ПС, которые оказались включенными в программы, появляются там «в готовом виде», вне всякой связи с предшествующим материалом курса. Тем самым предопределяется, что рассмотрение всех связанных с ними вопросов будет строиться в направлении от этих, неизвестным для учащихся образом появившихся, средств к поиску задач, которые с их помощью можно решить. Порочная логика! На практике мы всегда идем от потребностей к задачам, а от задач уже — к поискам средств и способов их решения.

Несмотря на то что в ныне действующей программе и упоминаются пакеты прикладных программ, в пробном учебном пособии это понятие полностью отсутствует. Настораживающая формулировка конкурсной программы «понятие о ППП как средстве автоматизации решения задач» в сопоставлении с § 17 II части пробного учебного пособия заставляет предположить, что это понятие имеется в виду представить учащимся в весьма обедненном виде — просто как совокупность готовых к выполнению программ.

Обе программы полностью обходят вопрос о системах автоматизации проектирования программного обеспечения. А ведь на самом деле большинство прикладных (да и не только прикладных) ПС будут создаваться именно с помощью таких систем, иным путем просто невозможно справиться с необходимым объемом работ. Зачем же это «засекречивать» от учащихся? Вряд ли такое умолчание о САПР программного обеспечения можно считать случайным, ибо оно тесно согласуется с упомянутой выше установкой авторов на предъявление учащимся конкретного «алгоритмического языка» в качестве основного якобы средства решения прикладных задач.

О применении вычислительной техники в народном хозяйстве: конкурсная программа требует лишь умения «приводить примеры применения ЭВМ в основных областях народного хозяйства, в повседневной деятельности человека»! Лишь в последнем его разделе упоминается о компьютерной грамотности. Там не раскрывается, какое именно содержание вкладывается в это понятие, однако обращение к приложению V второй части пробного учебного пособия заставляет предположить, что и здесь имеется в виду трактовка компьютерной грамотности как некоего абстрактного «умения программировать». А ведь на самом-то деле настоящая компьютерная грамотность (а лучше говорить здесь о компьютерной культуре) должна воспитываться на протяжении всего курса, а не сводиться к декларациям, отнесенным в самый его конец!

Успешность применения ВТ зависит и будет зависеть от готовности к этому многочисленной армии специалистов в конкретных предметных областях — инженеров, рабочих, служащих, научных работников, педагогов, руководителей разных рангов. Правильный выбор новых задач в соответствии с теми возможностями, которые открывает применение ЭВТ, их грамотная постановка, учитывающая как специфику конкретной предметной области, так и имеющиеся ресурсы, рациональная организация как самого их решения, так и использования получаемых результатов, способность осуществлять все это во взаимодействии со специалистами различного профиля — вот главное, что (в той или иной мере) требуется и будет требоваться от специалиста-«предметника». Всего этого никто другой за него сделать не сможет, потому что нельзя успешно ставить и решать задачи, не будучи специалистом в соответствующей области. А вот писать программы на том или ином алгоритмическом языке «предметнику» если (в очень редких случаях) и придется, то только самые элементарные. Изучение основ информатики ему нужно вовсе не для того, чтобы взять на себя функции программиста или электронщика, а ради возможности плодотворного с ними взаимодействия, в ходе которого каждый будет делать свою часть общего дела.

К сожалению, обе рассмотренные программы курса ОИВТ не только не дают возможности показать учащимся реальную проблематику «в целом», но и прямо нацелены на привитие им ошибочных представлений, которые потом будут серьезной помехой в практической деятельности. В свете всего сказанного выше рассмотрение чисто методических недостатков двух этих моделей курса представляется излишним. *Раз неверна концептуальная основа, значит, речь может идти только о создании принципиально новой программы.* Правомерно было бы, наверное, говорить даже не о программе, а о пакете программ, связанных между собой концептуальным единством и в то же время позволяющих учесть конкретные особенности, в которых проходит изучение курса.

Чтобы не ограничиваться лишь теоретическими рассуждениями о том, какими могут быть эти программы, приведем конкретный пример.

Программа курса «Основы информатики и вычислительной техники»

IX—X КЛАССЫ (102 ч)

Вариант В

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

Настоящий вариант ориентирован на следующее техническое обеспечение занятий:

а) для учащихся IX класса — учебный кабинет, оснащенный ПМК типа «Электроника Б3-34»;

б) для учащихся X класса — дисплейный класс на базе мини- или микро-ЭВМ или учебный кабинет, оборудованный персональными ЭВМ.

Курс предназначен для привития учащимся навыков использования вычислительной техники и вооружения их знаниями о роли и широком применении этой техники в народном хозяйстве. При его изучении используется материал, составляющий содержание курсов математики и физики. В свою очередь, знания, умения и навыки, полученные учащимися в процессе прохождения курса, используются при изучении других учебных дисциплин, при проведении факультативных занятий и во внеклассной работе.

В результате обучения учащиеся должны знать:

предмет информатики как научной дисциплины, ее современную роль и значение;

основные структурные компоненты средств ВТ, способы представления информации в памяти ЭВМ и на внешних носителях, общие принципы функционирования ЭВМ;

общие характеристики основных способов использования ЭВМ для управления техническими объектами;

основные команды и правила работы с ПМК типа «Электроника Б3-34»;

основные понятия и правила языка программирования Бейсик;

понятия математической модели, алгоритма, простейшие способы описания и представления алгоритмов;

общую характеристику современного программного обеспечения, средств и методов его создания и сопровождения;

основные направления применения ВТ в народном хозяйстве и их главные характеристики;

требования к компьютерной культуре современного специалиста.

Учащиеся должны уметь:

разрабатывать и описывать алгоритмы и программы решения на ЭВМ несложных задач и практически осуществлять такие решения с использованием системы команд ПМК и языка программирования Бейсик;

планировать и осуществлять решение задач с использованием уже существующего программного обеспечения и соответствующей документации;

взаимодействовать с вычислительной техникой в режиме диалога в процессе работы обучающих, контролирующих и т. п. систем.

СОДЕРЖАНИЕ ОБУЧЕНИЯ

IX КЛАСС — всего 34 ч

1. *Введение* (1 ч).

Что и зачем изучает информатика. Вычислительная техника. Краткая историческая справка. Понятие о роли и значении ВТ и ее применении в народном хозяйстве. Цели и задачи курса.

2. *Основные команды микрокалькулятора* (2 ч).

Что такое микрокалькулятор. Основные типы микрокалькуляторов, их функциональные возможности. Характеристика используемой модели микрокалькулятора. Порядок подготовки к работе и ее завершения. Структура микрокалькулятора.

Формы представления чисел, их ввод в микрокалькулятор. Начальные замечания о погрешностях.

Одноместные операции. Использование переключателя «радианы — градусы». Извлечение числа из регистра предыдущего результата (перемещение RX1→RX).

Последовательный ввод чисел в регистры операционного стека. Двухместные операции. Перемещение информации в операционном стеке при выполнении двухместных операций.

Запись чисел в регистры памяти и извлечение из них. Вычисления по формулам, включающим в себя несколько операций, с использованием операционного стека и регистров памяти.

3. *Составление простейших программ и их использование* (3 ч).

Программная память ПМК. Команды и коды. Правила ввода и корректировки программ. Ввод исход-

ных данных. Понятие о выполнении программы ПМК.

Организация отладки и счета по программе (в том числе использование пошагового режима, отладочных команд, промежуточных остановов и т. п.). Повторное использование программ.

Составление, отладка и выполнение программ для вычисления значений функций.

4. *Начальные понятия об алгоритмах* (2 ч).

Алгоритм как точное предписание о выполнении в определенном порядке некоторой системы операций, позволяющее, отправляясь от исходных данных, получить решение задачи. Ориентация алгоритма на конкретного исполнителя.

Примеры построения алгоритмов для решения несложных вычислительных задач с помощью ПМК.

Словесные описания алгоритмов. Описание алгоритмов с помощью блок-схем. Понятие о других способах описания алгоритмов. Используемые обозначения. Замечания по терминологии.

5. *Организация циклов и ветвлений* (5 ч).

Циклические участки алгоритмов, их описание. Использование «счетчика» для организации циклов с заданным числом повторений. Составление, отладка и выполнение программ для вычисления сумм, табулирования функций.

Ветвления в алгоритмах и их описание. Команды условного и безусловного перехода, их использование для организации ветвлений и циклов. Составление, отладка и выполнение программ для нахождения корней квадратных уравнений, нахождения наибольшего общего делителя и наименьшего общего кратного двух натуральных чисел, приближенного вычисления площади криволинейной трапеции (метод прямоугольников).

6. *Документирование отлаженных программ, их последующее использование* (3 ч).

Зачем и когда нужно документирование программ. Описание программы и инструкция по ее применению. Составление документации для программы нахождения наибольшего общего делителя и наименьшего общего кратного двух натуральных чисел. Использование ранее созданных и документированных программ для вычисления значения многочлена по схеме Горнера и для упорядочения по возрастанию или по убыванию последовательности чисел.*

7. *Подпрограммы* (3 ч).

Команды перехода к подпрограмме и выхода из нее, перемещения информации в адресном стеке указателя адреса. Способы обмена информацией между вызывающей и вызываемой программами. Условия, при которых целесообразно использование подпрограмм.

Составление, отладка и выполнение программ для приближенного нахождения изолированного корня уравнения (метод половинного деления), площади криволинейной трапеции (метод трапеций). Использование подпрограмм при решении задачи табулирования функций.

8. *Простейшая принципиальная схема решения прикладных задач с помощью ЭВМ* (3 ч).

Предварительный (опорный) пример. Понятие о математической модели. Основные этапы решения: построение математической модели, выбор метода решения, разработка структуры алгоритма, составление, отладка и выполнение программы, интерпретация и анализ полученных результатов. Иллюстративный пример.

Понятие о современных и перспективных подходах к решению прикладных задач с применением средств ВТ.

9. *Решение задач с конкретным (физическим, техническим и т. п.) содержанием* (9 ч).

Примерные задачи: нахождение табличной зависимости скорости падающего тела от времени (с учетом сопротивления воздуха). Определение мгновенной скорости в момент окончания движения по известным значениям пройденного пути и времени падения. Задача о движении тела, брошенного под углом к горизонту. Определение значения угла (при заданной начальной скорости) или значения начальной скорости (при заданной величине угла), необходимых для достижения требуемой дальности. Задача об опорожнении (наполнении) бассейна и опирающиеся на нее задачи. Исследование колебаний, происходящих под действием силы упругости и сил сопротивления. Исследование простейших популяционных моделей типа «хищник — жертва».

Нахождение табличной зависимости силы тока в цепи с самоиндукцией от времени (с момента замыкания цепи). То же — для содержания веществ при химических реакциях в растворах с перемешиванием и т. п. Решение задач, опирающихся на эти зависимости.

10. *Косвенные обращения к регистровой памяти* (2 ч).

Команды записи в память и извлечения из памяти с косвенным указанием адресуемого регистра. Модификация адресного указателя.

Одномерные числовые массивы. Описание алгоритмов, включающих в себя обработку одномерных массивов. Программирование обработки массивов (иллюстрация на примерах программ вычисления значений многочлена по схеме Горнера и упорядочения числовой последовательности). Составление, отладка и выполнение программы нахождения выборочного среднего значения, среднего абсолютного отклонения и максимального отклонения.

11. *Краткий обзор других возможностей ПМК* (2 ч).

Обращение к подпрограмме с косвенным указанием адреса. Условные и безусловные переходы с косвенным указанием адреса перехода.

Дополнительные команды старших моделей ПМК.

X КЛАСС — всего 68 ч

12. *Представление информации в ЭВМ* (3 ч).

Внутренняя память ЭВМ, ее виды, понятие об адресации. Двоичная система счисления. Представление различных типов данных во внутренней памяти ЭВМ. Понятие о типах и структуре машинных команд и об их представлении и размещении в памяти ЭВМ.

13. *Простейшая структурная схема ЭВМ* (3 ч).

Процессор. Устройство управления, арифметико-логическое устройство. Понятие об их структуре и функциональном назначении. Магистрали. Понятие об организации связи с внешними устройствами. Управление выборкой и выполнением команд.

14. *Внешние устройства ЭВМ* (3 ч).

Видеотерминал, его назначение, правила работы учащегося за видеотерминалом.

Устройство печати. Внешняя память.

15. *Простейшая схема функционирования ЭВМ* (2 ч).

Подготовительные операции. Загрузка программы, начальная установка указателя адреса команды, выполнение программы.

Понятие о прерываниях и их обработке.

16. *Характеристика современных средств ВТ* (3 ч).

Основные этапы развития ВТ. Понятие о поколениях ЭВМ. Обзор современных средств ВТ. Их классификация, перспективы развития.

Характеристика наиболее распространенных и перспективных типов ЭВМ.

17. *ВТ в управлении техническими объектами и системами* (3 ч).

Особенности организации связи с объектами управления. Понятие о цифро-аналоговых и аналого-цифровых преобразователях, устройствах индикации.

Встроенные микропроцессоры. Использование специализированных и универсальных ЭВМ.

Примеры систем автоматического и автоматизированного управления техническими объектами и системами.

18. *Введение в программирование на алгоритмическом языке Бейсик* (3 ч).

Предварительный пример. Символика языка. Общие правила записи программы. Формат операторов языка.

Числовые константы. Переменные. Арифметические выражения. Оператор прямого присваивания. Операторы INPUT и PRINT.

Понятие блока данных. Оператор заполнения блока данных и оператор извлечения значения из блока. Оператор RESTORE.

Стандартные функции и их использование.

Примеры программ.

19. *Работа с интерпретирующей системой Бейсик* (2 ч).

Назначение и сущность процедур компиляции и интерпретации. Понятие об интерпретирующей системе Бейсик.

Основные команды обращения к интерпретирующей системе. Ввод программы. Средства корректировки программы. Отладка программы.

Составление, отладка и выполнение простейших программ (вычисления по формулам).

20. *Ветвления и циклы* (4 ч).

Операторы безусловного и условного перехода. Организация в программе ветвлений и циклов с помощью оператора условного перехода.

Составление программ для нахождения значений «склеенных» функций, наибольшего общего делителя и наименьшего общего кратного двух чисел.

Организация циклов с помощью операторов FOR и NEXT.

Общие замечания об организации циклов.

Составление, отладка и выполнение программ для вычисления сумм, табулирования функций, приближенного вычисления определенного интеграла (метод средних прямоугольников).

21. *Массивы* (3 ч).

Понятие массива. Оператор описания размерности массива. Обращение к элементам массива.

Составление, отладка и выполнение программ нахождения наибольшего и наименьшего элементов массива, упорядочения одномерного массива по возрастанию или убыванию.

22. *Подпрограммы* (4 ч).

Понятие подпрограммы. Операторы обращения к подпрограмме и выхода из нее.

Составление, отладка и выполнение программ для приближенного вычисления корня уравнения (метод половинного деления), приближенного решения системы двух уравнений с двумя неизвестными (метод простых итераций), приближенного вычисления определенного интеграла (метод трапеций).

23. *Решение задач с конкретным* (физическим, техническим и т. п.) *содержанием* (10 ч) (типы задач аналогичны приведенным в п. 9).

24. *Практическая работа с конкретной диалоговой системой* (6 ч).

Назначение и функциональные возможности системы. Начало и окончание сеанса работы. Правила работы с системой, языковые средства пользователя.

Решение учебных задач с использованием диалоговой системы.

25. *Знакомство с проектированием и созданием использовавшейся диалоговой системы* (3 ч).

Характеристика круга задач, решаемых с использованием системы. Сравнительная оценка трудоемкости решения задач с использованием диалоговой системы и без нее.

Понятие об этапах проектирования, создания и ввода в эксплуатацию системы, о средствах, использованных при ее разработке. Квалификационная характеристика различных категорий специалистов, участвовавших в проектировании и создании системы и в ее эксплуатации.

26. *Эволюция средств создания программного обеспечения ЭВМ* (4 ч).

Непосредственная разработка машинного алгоритма: программирование в кодах команд ЭВМ. Языки программирования и трансляторы. Понятие о макропроцессорах. Пакеты прикладных программ и краткая характеристика программирования с их использованием.

Понятие об операционных системах.

Проблемы создания и сопровождения программного обеспечения. Понятие о современных средствах и технологии программирования.

27. *ВТ в народном хозяйстве* (8 ч).

Научно-технические расчеты — первая область применения ЭВМ. Современные средства автоматизации научно-технических расчетов. Характеристика соответствующих информационных процессов. Организация взаимодействия специалистов и сущность требований к их квалификации.

Информационные процессы организационного управления. Понятие о его автоматизации. Понятие о содержании и характере работ по проектированию, созданию и вводу в эксплуатацию соответствующих АСУ; характеристика деятельности различных категорий специалистов, организации их взаимодействия, требований к их подготовке. Особенности труда производственного и управленческого персонала в условиях функционирования АСУ.

Автоматизация управления техническими объектами и системами. Характеристика информационных процессов этого управления.

Понятие об организационно-технологических АСУ. Автоматизация конструкторского и технологического проектирования.

Автоматизированные информационно-поисковые системы. Понятие о банках данных. Характеристика областей применения, особенностей проектирования, соз-

---

* В этих программах умышленно используются еще не известные учащимся команды косвенного обращения к памяти.

дания и эксплуатации автоматизированных информационных систем.

Перспективные направления использования средств вычислительной техники, их основные характеристики.

28. *Информатика и ВТ — ключевое звено научно-технического прогресса* (4 ч).

Информационные ресурсы, их роль и значение на современном этапе научно-технического прогресса.

Индустрия информатики, перспективы ее развития. Требования к подготовке современного специалиста.

Программа эта разработана автором еще в 1985 г. по настоянию учителей, проходивших обучение в Ленинградском институте методов и техники управления (ЛИМТУ), и успешно опробована на практике.

Две другие программы, составляющие с ней единый «пакет», ориентированы на иные варианты использования в учебном процессе средств ВТ.

Прокомментируем кратко предусматриваемую ей логику построения курса.

Следующие за кратким (больше одного урока отводить на него нет смысла) введением темы 2 и 3 посвящены первоначальному освоению работы с ПМК в «ручном» и «автоматическом» режимах. Отметим, что изучение этих тем с самого начала идет в тесной связи с рассмотрением структуры самого ПМК и логикой взаимодействия его компонентов. Это не только помогает достичь более глубокого усвоения материала, но и благотворно сказывается при последующем изучении тем, относящихся к структуре и функционированию средств ВТ. В теме 3 появляются и начальные понятия об отладке программ.

Понятие алгоритма впервые появляется лишь в теме 4. Очень важно, что происходит это на фоне естественной для учащихся мотивации — необходимости спланировать ход вычислений в тех случаях, когда непосредственное составление машинной программы начинает вызывать затруднения. Учащиеся сначала знакомятся со словесным описанием структуры алгоритма, а затем начинают использовать блок-схемы. Им дается понятие (не более того) о некоторых других способах описания структуры алгоритма, но в дальнейшем преимущественно используются блок-схемы. Это наиболее простой и легко воспринимаемый учащимися способ, не создающий к тому же никаких препятствий при дальнейшем прохождении курса.

В темах 5 и 7 продолжающееся изучение вопросов алгоритмизации и программирования (ветвления, циклы и подпрограммы) происходит параллельно, что гораздо лучше отвечает психологическим закономерностям усвоения материала, чем последовательное изучение сначала алгоритмических структур и лишь затем соответствующих вопросов программирования.

Одной из причин, обусловивших как само включение в программу темы 6, так и близкое к началу курса ее размещение, явилось стремление как можно раньше обеспечить возможность действенного применения ПМК при изучении других дисциплин. Без обращения к готовым программам сколько-нибудь широкое использование калькуляторов за пределами курса информатики просто невозможно из-за больших затрат времени на составление и отладку программ.

Узловыми для курса IX класса являются темы 8 и 9. Задачи, рассматриваемые в теме 9, скомпонованы в отдельные группы таким образом, чтобы решение последних задач каждой из этих групп сводилось к небольшому вычислительному эксперименту. В процессе изучения этих тем внимание учащихся специально обращается на проявляющиеся даже на уровне учебных задач характерные аспекты, связанные с применением ВТ. Так, достаточно наглядным и убедительным выглядят для учащихся следующие положения:

ВТ кардинально расширяет человеческие возможности, позволяя решать *совершенно новые классы задач* (неравноускоренное движение, динамика популяций и т. д.);

способы этих решений, как правило, *кардинально отличаются* от тех, которые не опираются на использование ЭВМ (пошаговое моделирование процессов, вычислительный эксперимент вместо расчетов по «конечной формуле»);

решение задач невозможно без взаимодействия со специалистами конкретной предметной области (задачи, как правило, будут требовать обращений учащихся к преподавателю за консультацией по поводу конкретных физических, химических, биологических и т. п. закономерностей, не известных им из курсов соответствующих дисциплин).

Курс X класса начинается с изучения раздела, посвященного структуре и функционированию средств вычислительной техники. По-видимому, особых пояснений темы 12—17 не требуют.

Изучение основ программирования на алгоритмическом языке Бейсик (выбор его обусловлен конкретными условиями проведения занятий) начинается с вводной темы 18. При ее прохождении учащиеся, широко используя демонстрационные примеры, получают необходимые начальные представления как о самом языке, так и о составлении и выполнении простейших программ. Только на такой «ориентировочной основе», включающей в себя еще и материал тем 12—15, оказывается продуктивным разъяснение им сущности процедур компиляции и интерпретации (разумеется, без каких бы то ни было излишних подробностей!). Темы 20—22 завершают знакомство учащихся с языком, которое, как можно видеть, ограничивается необходимым минимумом его средств.

Тема 23 представляет собой своего рода логическое продолжение изучавшихся в IX классе тем 8 и 9. Существенно большие по сравнению с использованием ПМК удобства программирования на языке Бейсик позволяют не только увеличить количество и повысить уровень сложности решаемых задач. Появляется возможность манипулирования фрагментами программ, неоднократного их модифицирования при изменении постановки задачи. В определенной мере это облегчает учащимся изучение некоторых вопросов тем 25 и 26.

Работая с конкретной диалоговой системой (тема 24), учащиеся получают прекрасную возможность как наглядно сопоставить удобства ее использования по сравнению с ранее изученными ими подходами к решению прикладных задач, так и прочувствовать реально существующие ограничения. Тема 25, изучение которой прочно опирается на полученный учащимися опыт работы с системой, дает им возможность на конкретном материале познакомиться с тем, как проектируются и создаются сложные программные комплексы. Она, в свою очередь, является своеобразным «мостиком» к достаточно трудной как для преподавателя, так и для учащихся теме 26.

Завершается курс темами 27—28. Именно в процессе их изучения у учащихся окончательно формируется система знаний о том, как применяется ВТ в различных областях народного хозяйства, каковы ее роль и значение на современном этапе научно-технического прогресса, какие требования предъявляются к специалистам, использующим (прямо или косвенно!) эту технику, как должны они взаимодействовать с ней и друг с другом. Как можно видеть из текста программы, имеется в виду достаточно обстоятельное раскрытие этих тем. Предшествующий материал (особенно темы 8—9, 23—26) дает необходимые для этого предпосылки. Но надо иметь в виду, что здесь требуется и высокий уровень подготовки самого учителя.

В этой связи следует отметить, что регулярно проводимая учеба учителей информатики и необходимость осуществления ими большой самоподготовки ложится на них громадной дополнительной нагрузкой, которая во многом не планируется и не учитывается. В интересах дела — возможно скорее исправить такое положение вещей, даже если для этого потребуются нестандартные решения. *Уровень и характер подготовки учителей — это фактор, влияющий на выработку реально осуществимой программы курса в не меньшей степени, чем наличие или отсутствие тех или иных технических и программных средств.*

Досадно, конечно, что при реализации представленной здесь модели курса много времени затрачивается на чисто технические аспекты программирования в системе команд ПМК. Думается, что по крайней мере вопросы, составляющие темы 2 и 3, вполне могли бы изучаться в курсе математики VIII класса.

Иногда подвергается сомнению целесообразность использования ПМК в школьном курсе информатики. Согласиться с этим в настоящее время нельзя. Во-первых, по сравнению с «безмашинным» построением курса, преимущества, связанные с применением этих средств, не только теоретически очевидны, но и получили убедительное подтверждение на практике. Во-вторых, только на такую ВТ и может сейчас реально рассчитывать массовая школа, имея в виду сколько-нибудь широкое использование ее не только в курсе информатики, но и в других дисциплинах. Представляется необходимым принять все меры к скорейшему обеспечению школ ПМК.

В рассмотренном варианте программы явным образом не представлены такие средства, как текстовые редакторы и системы машинной графики. Это объясняется прежде всего большим разнообразием используемых сейчас в учебном процессе типов ЭВМ и недостаточностью имеющегося на многих из них программного обеспечения. Знакомство учащихся с указанными средствами, разумеется, не исключается в тех случаях, когда это возможно. С другой стороны, в настоящее время для небольшого общеобразовательного курса эти средства нельзя признать и безусловно необходимыми.

В заключение отметим, что *ни представленная здесь, ни какая-либо другая программа не может претендовать на сколько-нибудь длительный срок существования.* Во-первых, уже в ближайшие годы отнесение начала изучения информатики в средние классы окажется и возможным, и необходимым. Во-вторых, действительная компьютеризация обучения не может ограничиваться рамками одного только этого курса, а требует широкого и систематического применения ВТ во всем учебно-воспитательном процессе, а в таких условиях и курс информатики должен будет стать совершенно иным. В-третьих, выдвигать новые требования к курсу будут новое техническое оснащение школ, развитие и повышение доступности различного рода информационных систем.

Нельзя забывать, что курс этот, сильно отличающийся от традиционных школьных дисциплин, находится еще в самом начале своего становления. Поэтому очень важно на практике опробовать различные его модели, тщательно изучить и обобщить накапливаемый опыт. *Каждый заслуживающий внимания подход к построению курса должен встречать «режим благоприятствования», обеспечиваться изданием учебных пособий, тиражированием необходимого программного обеспечения, методической и организационной поддержкой.*

А. ИГЛИЦКИЙ, Е. КРЫЛОВА, В. ЛУЦКИЙ, М. ПЕТРОВ

Рыба ли рак?

«Видно, что повар руководствовался более каким-то вдохновением и клал первое, что попадалось под руку... было бы горячо, а вкус какой-нибудь, верно, выйдет».

Н. В. Гоголь

В № 2 журнала «Информатика и образование» за 1986 г. была опубликована программа специального 36-часового курса подготовки руководящих и педагогических кадров народного образования по ОИВТ и сообщалось: «Разработка программных средств для этого и методических рекомендаций по их применению (для персональных ЭВМ «Агат», «Ямаха», ДВК-2М и др.)... завершится к концу года», «активное обучение слушателей по данному курсу начнется со II квартала 1987 г.». В частности, разработкой программно-методического комплекса для «Агата» занимался НИИ ШОТСО АПН СССР.

Каковы же результаты этой работы?

«Сбор грибов» — так можно охарактеризовать деятельность работников лаборатории ИВТ НИИ ШОТСО*. Программное обеспечение к «Агату» беспрецедентно по своей бедности, и программная часть так называемого программно-методического комплекса представляет собой просто механическое собрание разнородных программ, и без того имеющихся практически у всех пользователей (и даже частично ими созданных — о качестве этих программ говорить не приходится).

Создавшееся положение с программным обеспечением ПЭВМ «Агат» (а также других типов ЭВМ, установленных в школах) вообще ненормально и дискредитирует саму идею использования ЭВМ в школе. Оно вызвано изначальным отсутствием единой школьной ЭВМ, а также отсутствием стратегической линии в разработке соответствующего программного обеспечения.

По поводу предложенной НИИ ШОТСО методики проведения курса и его программной поддержки можно заметить следующее:

1. Тематическое планирование, предложенное НИИ ШОТСО, отличается от опубликованного в № 2 журнала «Информатика и образование» за 1986 г. (вместо 6 ч лекций — 11, общее количество часов сокращено до 34).

2. Предложенные программы реализованы в различных операционных системах: ДОС 3.3 и ППП «Школьница». Кроме того, система подготовки текстов «Агат—автор» и система численного моделирования являются полностью автономными. Таким образом, имеются 4 попарно несовместимые возможности использования ПЭВМ «Агат», что дезориентирует даже подготовленного пользователя.

3. В предложенный комплекс в соответствии с разделами тематического планирования помимо систем «Агат — автор» (разделы 3, 4) и СЧМ (раздел 5) вошли программы:

раздел 1 —
Теннис (игра, ДОС 3.3);
Дорога (игра, ДОС 3.3);
Шашки (игра, ДОС 3.3);
Шахматы (игра, ДОС 3.3);
Кубик Рубика (игра, ДОС 3.3);
Отелло (игра, ДОС 3.3);
Anakonda (игра, ДОС 3.3);
раздел 2 —
Дежурик (исполнитель, «Школьница»);
Муравей (исполнитель, «Школьница»);
Шлюзик (исполнитель, «Школьница»);
KLAWI (тренажер, ДОС 3.3);
раздел 6 —
АИСТ (архивно-информационная система, «Школьница»);
раздел 7 —
MEDIC (медицинский тест, ДОС 3.3);
раздел 8 —
Лучи и спираль (демонстрационная программа, ДОС 3.3);
GR256 (графический редактор, ДОС 3.3);
DEM (графический редактор, ДОС 3.3);
NEWGR (графический редактор, ДОС 3.3);
GREDIT (графический редактор, ДОС 3.3);
раздел 9 —
TEXT (ППС — педагогическое программное средство, ДОС 3.3);
Химия (ППС, «Школьница»);
Химлаб (ППС, «Школьница»);
Русский язык (ППС, «Школьница»);
Сернокислот (1—6) (ППС, «Школьница»);
Геометрия (1—3) (ППС, «Школьница»);
MULTA (ППС, ДОС 3.3)**
WORDS (тренажер, ДОС 3.3).

4. Программа спецкурса (по своему замыслу) призвана:

обеспечить знакомство руководящих и педагогических работников страны с основами информатики;

показать пути использования ИВТ на уроке, во внеклассной и внешкольной работе, в управлении учреждениями системы просвещения.

Представленный программно-методический комплекс, названный «Информатика-87», был апробирован на курсах для учителей одной из школ Москвы летом 1987 г.

О соответствии представленного набора программ целям и задачам каждого из разделов спецкурса можно сказать следующее.

Раздел 1. Знакомство с ПЭВМ «Агат». Для поддержки этого раздела предложены компьютерные игры.

Педагогическая ценность игр вообще сомнительна. Вопреки распространенному мнению, они отнюдь не помогают устранить психологический барьер в общении неподготовленного взрослого человека с компьютером. Что касается данных игр, то они либо слишком примитивны (Теннис, Дорога, Anakonda), либо, напротив, чересчур интеллектуальны (Шашки, Шахматы, Отелло, Кубик Рубика) и превращаются в самоцель, не имеющую отношения к компьютерной грамотности. Кроме того, качество игр самостоятельной разработки (кроме последних четырех, адаптированных с «Apple») оставляет желать лучшего.

Раздел 2. Исполнители. Примеры исполнителей. Системы команд исполнителей. Он изучается вместе с предыдущим в ходе первого занятия. Предложенное программное обеспечение предполагает загрузку всех машин (которые полностью автономны и не объединены в сеть) сначала ДОСом, потом «Школьницей», что представляет слишком большую сложность для неподготовленного пользователя. Сами же модели исполнителей достаточно адекватно воплощают идею формального выполнения команд в диалоговом режиме. Однако следует отметить, что уровень этих моделей и мотивация пользователя соответствует примерно развитию первоклассника, для которого программы и были написаны. У взрослых пользователей данные исполнители вызывают лишь легкое умиление, которое сменяется недоумением. Что же касается клавиатурного тренажера KLAWI, то он, естественно, не является моделью исполнителя и более уместен при первичном знакомстве с ЭВМ, в разделе 1. Правда, эта программа скучновата, лучше было бы для первичного знакомства использовать программу WORDS.

Разделы 3—4. Обработка текстов. Данные разделы предполагают работу с системой «Агат—автор» (адаптированная Apple Writer). Он является вполне завершенным текстовым редактором, который может быть использован даже в профессиональной редакционно-издательской деятельности. Однако именно в силу профессиональной ориентации он не совсем удобен для обучения, хотя, по мысли авторов комплекса, предназначен «... для составителя текстов, имеющего средний любительский навык работы на пишущей машинке». Опрос учителей, занимавшихся на курсах, показал, что такой навык скорее исключение, чем правило. Кроме того, работа с этим текстовым редактором (ввиду некоторых неудобств, присущих «Агату») уже предполагает определенный уровень «компьютерного» мышления, т. е. той самой компьютерной грамотности, которую еще надлежит сформировать.

Раздел 5. Электронные таблицы. Система численного моделирования (СЧМ) является точной копией системы VisiCalc; к ней с небольшими изменениями относится все сказанное о системе «Агат—автор». Хочется оспорить утверждение составителей комплекса о том, что «...работа с данной программой не требует никаких навыков программирования»,— работа с СЧМ представляет собой именно программирование на языке макрокоманд данной системы.

Раздел 6. Информационные системы. Здесь предполагается работа с архивно-информационной системой «АИСТ». Как показала практика, она вполне удовлетворяет целям курса, но ее использование сильно усложняет уже упомянутый разнобой в средствах программирования.

Раздел 7. Здравоохранение. Программа MEDIC — диалоговая программа опроса состояния здоровья — вызвала живой интерес слушателей, что наводит на мысль о большой целесообразности использования ее в первом разделе для снятия пресловутого «барьера». Все же признать эту программу полноценным программным продуктом мешает отсутствие сопровождающей документации и расшифровки диалога на экране. Разумеется, только врач достаточно компетентен для оценки адекватности результатов диалога пациента с компьютером, но результаты опроса слушателей при помощи данной программы оказались чересчур неожиданными для большинства опрошенных; это вызывает сомнения в правильности разработки сценария. Кроме того, одной программы MEDIC явно недостаточно для работы по разделу 6 в соответствии с планированием (2 ч).

Раздел 8. Основные понятия программирования. Команды ветвления и повторения. Машинная графика. Для преподавания раздела 8 предлагается текст лекции, в основном повторяющий концепцию изложения соответствующего материала в пробном учебном пособии по ОИВТ для IX класса. Уже не говоря о том, что само это пособие вызывает много нареканий методического характера, оно рассчитано на школьников, а спецкурс рассчитан на взрослых людей, которых, очевидно, надо учить иначе. В частности, так как на «Агате» не реализован язык Лого, обучение машинной графике в этом изложении оказывается лишено программной поддержки. Далее, кодирование алгоритма, разработанного в алгоритмической нотации, на производственном языке программирования, в частности на Бейсике, представляет собой особую методическую проблему. В предложенном теоретическом материале она совершенно не проработана.

Предлагаемый методический прием изучения конструкций Бейсика путем редактирования готовой программы в силу специфики ПЭВМ «Агат» оказывается труднореализуемым, так как для сознательного редактирования программ на «Агате» требуется умение работать с буфером входной строки. Возможно, на данном типе ПЭВМ лучше предложить слушателям написание маленьких собственных программ с их последующей модификацией, а также разбор готовых текстов чужих программ, но без редактирования и модификации.

Для обучения созданию графических образов на экране компьютера предлагается четыре (!) практически однотипных, отличающихся только степенью разрешения и количеством цветов, графических редактора***. Самый удачный из них, NEWGR, требует предварительных загрузки другого редактора и объявления графической страницы. Он не представляет собой законченного продукта и абсолютно не документирован (так как автор писал эту программу не для обучения и не планировал ее сопровождения другим пользователем), не позволяет программировать получение изображений и предполагает рисование «вручную», а действия, связанные с сохранением и последующим использованием графической информации, требуют достаточно хорошего знания архитектуры ПЭВМ «Агат». Фактически графические редакторы, представленные в этом разделе, являются неплохими примерами исполнителей и были бы более уместны в разделе 2.

Таким образом, раздел 8, являясь одним из самых важных в курсе, оказался наименее подготовленным как теоретически, так и практически.

Раздел 9. Учебный процесс. Этот раздел должен являться логическим завершением всего обучения на курсах, создать необходимые предпосылки для осознанного использования учителями ЭВМ в учебном процессе и решении задач управления на уровне школы.

Предложенная по данному разделу лекция фактически является конспективным изложением статьи И. Роберт «Какой должна быть обучающая программа?», опубликованной в «Информатике и образовании» № 2 за 1986 г. и предназначенной для разработчиков сценариев ППС,

---

* На момент написания настоящей статьи собранный НИИ ШОТСО «урожай» занимает 6 дисков: а) игры; б) текстовый редактор; в) программы для специальных целей (копирование и т. п.); г) комплекс программ Павлодарского педагогического института (ничего особенного, но сделан грамотно, что для НИИ ШОТСО пока недостижимо); д) обучающие программы по Бейсику Московского радиоаппаратостроительного техникума; е) десятка полтора демонстрационно-иллюстративных программ.

** Не путать с программой-мультипликатором Ветютнева.

*** Внимание! Программа GREDIT содержит ошибку: при попытке записи изображения на магнитную ленту диск затирается!

а не для формирования общего представления о многообразных возможностях использования ЭВМ в средней школе. В результате чтения этой лекции не был снят главный вопрос учителей: «для чего же нужен компьютер в школе?»

Программная поддержка этого раздела недостаточна как по объему, так и по заложенным в ней педагогическим и дидактическим идеям. Практически все предложенные ППС имеют те или иные существенные недоработки по сценарию (исходя из вышеупомянутой статьи И. Роберт) и реализации; в значительной мере это относится к программам MULTA и Геометрия. Вызывает большие сомнения педагогическая целесообразность использования на завершающем этапе обучения программы TEXT, так как она не имеет разумного применения и, следовательно, здесь даже вредна. Однако эта же программа была бы полезна при изучении раздела 1. Программы, входящие в ППП «Школьница», более удачны как по сценарию, так и по разработке, но не всегда защищены от случайных действий пользователя, что приводит к сбоям, а повторный их запуск требует знания операционной среды (следует отметить, что «Школьница», в отличие от Бейсика, вообще крайне плохо переносит «нештатные» ситуации).

Результаты проведенных курсов позволяют сделать следующие выводы:

в обучении учителей достигнута только первая из перечисленных педагогических целей, т. е. первичное ознакомление с ЭВМ; психологический барьер в отношениях учителей с компьютером снят;

ни один из учителей не вынес целостного представления о реальном практическом применении ЭВТ в школе;

частичный успех курсов обусловлен наличием отдельных удачных программ;

в целом предложенный программно-методический комплекс «Информатика-87» не соответствует педагогическим целям программы специального 36-часового курса подготовки руководящих и педагогических кадров народного образования по ОИВТ и малопригоден для дальнейшего использования.

Авторами данной статьи на основании проведенных ими курсов для учителей был написан и сдан в НИИ ШОТСО отзыв об этом комплексе (в него вошли практически все вышеприведенные замечания). Однако, ссылаясь на несоблюдение формы (которое действительно имело место, хотя и не по вине авторов), НИИ ШОТСО игнорировал нежелательную для него оценку.

В заключение, возвращаясь к заголовку статьи, авторы хотят со всей решительностью заявить, что даже на безрыбье рак не должен считаться рыбой. Существует реальная опасность того, что другие коллективы институтов АПН СССР могут при разработке программно-методических комплексов пойти таким же путем, но это снова приведет к плачевному результату. А пострадают от этого не «повара», которые готовят такое странное блюдо, а те, кому придется его есть. Лозунг «Информатика — любой ценой» уже достаточно себя скомпрометировал, и вопрос о качественном программном обеспечении должен быть решен безотлагательно.

А потом она скажет, что она думает о вас...

ЧТО МОЖЕТ ЭВМ?

«Искусственный интеллект», как его понимают авторы соответствующих исследовательских программ, совсем не похож на интеллект в обычном понимании этого слова. Программы «искусственного интеллекта», хотя и открывают новые возможности использования ЭВМ, не ставят целью моделирования мышления человека.

Но существуют уже и другие электронные машины — не вычислительные, а «думающие». Их называют нейронными сетями. Конструктивно они напоминают мозг, включают множество электронных «нейронов», сделанных из микропроцессоров, и сложнейшую систему их связей — едва ли не «каждый с каждым».

Способности такого нейросетевого компьютера весьма необычны. Он не может умножить 2 на 2, зато совершить ошибку ему ничего не стоит. Бессмысленная игрушка? Нет. Именно на пути проб и ошибок лежат его достижения.

На выходе нейросети поставили динамик и устройство, сравнивающее издаваемые ею звуки со звучанием английских слов, т. е. «сообщающие» ей, насколько близко к английскому произношению ее «бормотание». И сеть стала учиться говорить.

Первоначально ячейки этой машины были заполнены случайным содержимым, попросту говоря «мусором», и издаваемые звуки были вполне бессмысленными. Но уже через час в них появились паузы, напоминающие перерывы между словами. Самообучение продолжалось, и через сутки нейросеть почти безошибочно «говорила» по-английски.

Младенец научился говорить — а сколько еще у него впереди?!

Нам пишут

В. ЛОГАШЕВ, В. ВЫСОКОДВОРСКИЙ, Ю. КАЗАНСКИЙ

Робот для школьника

Специализация пользователей ЭВМ будет осуществляться по многим направлениям, но, на наш взгляд, наибольшую группу составят пользователи автоматизированных систем управления объектами, в которых ЭВМ является универсальным управляющим средством.

На сегодняшний день школьная программа курса основ информатики и вычислительной техники предусматривает лишь теоретическое ознакомление с АСУ. На практических занятиях учащиеся не получают соответствующих знаний, так как они составляют и реализуют учебные программы, не носящие управляющего характера или оперируют математическими моделями явлений, процессов, зачастую забывая о предназначении этих моделей. Конечный результат выполнения таких программ — алфавитно-цифровая графическая информация на экране дисплея или на бумажном носителе.

На наш взгляд, этого недостаточно. «Материализация» ученических программ является обязательным компонентом компьютерного всеобуча.

Знакомство с автоматизированными управляющими системами на базе ЭВМ нужно осуществлять не только на уровне их математических моделей, но и на реальных объектах, в качестве которых могут выступать как образцы промышленного оборудования (роботы, станки с ЧПУ и т. д.), так и их модели. Поэтому целесообразно создание дополнительного периферийного оборудования для школьного компьютера в виде различных кибернетических игрушек или компактных действующих моделей, прототипами которых могут послужить современные роботы или другие устройства, управляемые с помощью ЭВМ.

Подобное оборудование используется в профессиональном образовании и на факультативах в школах нашей страны и за рубежом.

...В ленинградском СПТУ-38 создали мастерскую-лабораторию. Манипуляторы различных систем здесь можно имитировать на специальных выносных панелях.

...Болгарские кибернетики представили математическим спецшколам страны первые мобильные учебные роботы с элементами искусственного интеллекта: они могут синтезировать речь, анализировать звуковые и световые сигналы. Такие роботы, названные «Робко-01», можно подключать к персональным компьютерам «Правец». С этим роботом уже работают и советские школьники в кружке «Юный программист» на Центральной станции юных техников Министерства просвещения РСФСР.

...В одной из английских школ введен экспериментальный курс робототехники. В ходе занятий школьники сами конструируют роботы и программируют их действия.

...Итальянской фирмой «Электроника Венета» выпускаются различные обучающие роботы. В том числе мини-робот RB-4/E, который вместе с управляющим компьютером умещается на столе. Этот робот представляет собой манипулятор с шестью степенями свободы. Предназначен для получения элементарных знаний о роботах, приобретения навыков написания управляющих программ, проведения несложных опытов*.

Правомерно ли ставить вопрос о широком оснащении школ такими моделями именно сейчас и не является ли подобное оборудование привилегией лишь тех учебных заведений, где ведется соответствующая профессиональная подготовка?

Мы считаем, что возможность проверить управляющие программы в реальной работе на практических занятиях в школе намного повысит эффективность обучения.

Школьные роботы нужно создавать именно сейчас, на этапе разработки и опытной эксплуатации различных образцов школьных компьютеров.

Они должны обладать достаточной универсальностью, многофункциональностью использования, при этом быть дешевым, построенным по модульному принципу, компактным, простым и безопасным в эксплуатации учебным оборудованием.

Следует отметить, что в моделях проще реализовывать новейшие идеи робототехники, ориентируя учеников на оборудование завтрашнего дня.

В идеале требуется создать набор различных

* Эти примеры, взятые нами из газет «Смена», «Учительская газета», журналов «Наука и жизнь» и «Техническая эстетика», можно продолжить.

моделей: манипулятор, транспортный робот, министанок и т. п. Такой комплект с успехом использовался бы в лабораториях ПТУ, техникумов и высших учебных заведений.

Однако сегодня более реальным представляется создание одной модели, обладающей достаточной универсальностью, простотой реализации и широкими методическими возможностями.

Желательно, чтобы модель выполнялась в групповом и индивидуальном вариантах различаясь лишь размерами и ориентацией в пространстве (групповой вариант — вертикальное расположение, обеспечивающее фронтальный показ; индивидуальный — горизонтальное).

В настоящее время авторы данной статьи на основе сформулированных ими педагогических и технических требований начали конструкторскую проработку универсального стенда, который предполагается использовать в общеобразовательных школах на уроках «Основы информатики и вычислительной техники», в лабораториях робототехники средних специальных учебных заведений.

Отличиями этого изделия от уже существующих будут универсальность его применения в учебном процессе, более широкий диапазон выполняемых функций.

Расположенный вертикально стенд имеет рабочее поле 1000×1000 мм. На нем размещены подвижные каретки. Поле выполнено на основе линейных шаговых двигателей. Подача импульса в обмотку двигателя заставляет каретки перемещаться на заданный шаг по координате X или Y. Таким образом, каретки могут быстро и точно передвигаться по любой заданной программой плоской траектории.

На рабочем поле располагаются различные предметы, удерживаясь на поверхности при помощи постоянных магнитов. Эти объекты могут механически перемещаться движущейся кареткой. Система датчиков фиксирует их координаты, текущее значение которых передается ЭВМ. Такой подход к построению модели делает ее широкоуниверсальной, так как на рабочем поле стенда и на каретках можно размещать различные объекты (модели или макеты станков, инструменты, материалы, муляжи животных, куклы, декорации и т. д.), взаимодействие которых определяется разрабатываемой учащимся программой.

Интересным и полезным с методической точки зрения может стать например, «оживление» при помощи модели исполнителей языка Робик — МУРАВЬЯ и МАШИНИСТА.

Применение в школе подобных моделей, основанных на различных конструктивных принципах, поднимет на более высокий уровень качество преподавания курса основы информатики и вычислительной техники, в будущем поможет молодым людям быстрее адаптироваться к реальным условиям современного производства.

Безусловно, такие устройства найдут применение и в других учебных заведениях.

С. ОСИПОВА, В. ЧЕРНОВ

Взаимная заинтересованность

К настоящему времени вузы страны накопили значительный опыт использования ЭВМ в учебном процессе. Они могут оказать школе существенную помощь в преодолении трудностей, связанных с компьютеризацией обучения. Вузы могут предоставить возможность учителям освоить работу на ЭВМ, поделиться со школой методическими приемами обучения, проводить лабораторные занятия на базе институтского ВЦ, организовать научные кружки по профилю вуза, с использованием на занятиях вычислительной техники, т. е. проводить эффективную профориентацию, найти своего абитуриента.

В г. Красноярске при Кировском отделе народного образования создан совет по информатике, объединяющий методистов, учителей, преподавателей учебно-производственного межшкольного комбината района, а также преподавателей Красноярского института цветных металлов им. М. И. Калинина. Задача совета — координация и совершенствование в школах учебно-методической работы по изучению и преподаванию курса информатики и вычислительной техники. Он контролирует использование ВТ в учебном процессе, рассматривает и распределяет по школам методические материалы. По инициативе совета, под руководством ученых Красноярского института цветных металлов проводился методический семинар учителей, на котором изучались основы алгоритмизации и программирования, а также рассматривались методические вопросы проведения лабораторных занятий на ПЭВМ «Агат».

В институтском вычислительном центре школьники разработали программы различных игровых ситуаций, а также контролирующую программу, одобренную методическим семинаром учителей и принятую к внедрению для контроля знаний учащихся.

При кафедре высшей математики Красноярского института цветных металлов несколько лет работает кружок «Юный программист» для учащихся VIII—X классов. Часть школьников района проходит летнюю производственную практику по специальности «оператор ЭВМ». Советом по информатике проведена районная заочная олимпиада, в которой приняли участие школьники IX—X классов.

Результаты исследовательских работ, алгоритмы и программы, составленные школьниками, были представлены на краевой конференции по информатике и вычислительной технике, проведенной Красноярским государственным университетом.

От активности, взаимной заинтересованности средней и высшей школ, согласованности их действий в большой степени зависит успех компьютеризации обучения.

Информация

Компьютеры на «Электро-87»

Современное электротехническое оборудование не было бы современным без вычислительной техники, которая повышает эффективность и надежность работы различных электродвигателей, генераторов, сварочной и осветительной аппаратуры, высоковольтных преобразователей энергии и других электротехнических устройств. Без компьютеров не могут работать робототехнические комплексы и гибкие автоматизированные производства, внедряемые сейчас в электротехнической промышленности. Ее достижения были показаны на Международной выставке электротехнического оборудования и линий электропередач «Электро-87», проходившей в Москве в июле этого года. На выставке свою продукцию, среди которой было и оборудование, оснащенное микропроцессорами, демонстрировали более 600 фирм, предприятий и объединений из 23 стран мира. Среди них — Австрия, Великобритания, Италия, США, Франция, ФРГ, Япония.

Достижения социалистических стран в совместном производстве электротехнического оборудования представлены в экспозиции международной организации «Интерэлектро» и на стендах НРБ, ГДР, ПНР, СРР, ЧССР.

Планами «Интерэлектро» предусмотрено создание современных электроприводов для станков с числовым программным управлением, многие из которых оснащены микропроцессорами. Только в СССР с такими приводами освоен выпуск более 100 моделей металлорежущих станков с ЧПУ. Эти приводы позволили увеличить производительность станков на 20—30 %, улучшить качество обрабатываемых деталей.

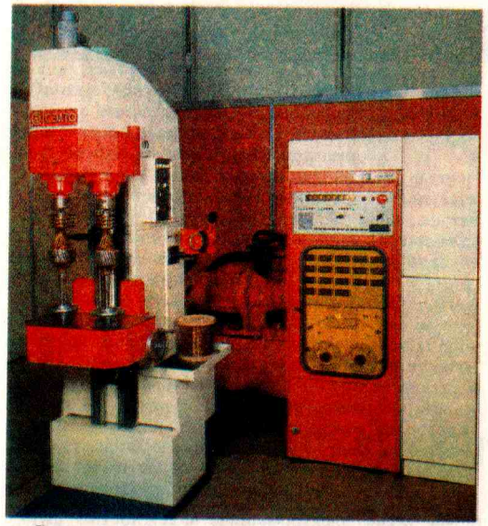
На выставке советская экспозиция была самая представительная. Здесь демонстрировались макеты мощных гидро- и турбогенераторов, разнообразные электроприводы, лазерно-технологические комплексы, осветительная аппаратура, медицинская, бытовая электротехника и другое оборудование.

Единичная мощность современных турбогенераторов приближается к миллиону киловатт. Только с помощью ЭВМ можно следить за работой таких электромашин, надежностью функционирования их механических частей, температурным режимом, электрическими параметрами статора и ротора.

Для автоматизации управления турбогенератором может использоваться показанный на выставке комплекс микросредств управляющей вычислительной техники МСУВТ В7/В9. Основными его элементами являются многослойные печатные платы с микропроцессорами. Этот комплекс отличается большими вычислительными управляющими возможностями, оригинальными системами связи, обеспечивает диагностику неисправных схем. Предусмотрено наращивание вычислительной мощности комплекса в широких пределах и подключение разнообразных устройств, что позволяет формировать распределительные управляющие вычислительные сети.

Широкое использование электроники и микропроцессоров способствовало расширению производства печатных плат на электротехнических предприятиях. Здесь компьютеры используются в стадии проектирования печатных плат и при их серийном выпуске. Посетители выставки могли познакомиться с новой системой «Луч-12», созданной для проектирования фотошаблонов печатных плат. Эта система с отображением информации на экране дисплея позволяет работать на формате поля кодирования размером 960×640 мм, при шаге координатной сетки 5 мм. Оператор может использовать в своей работе 64 цветовых оттенка. В процессе проектирования фотошаблона информация кодируется и составляется программа для управления исполнительными устройствами технологической линии.

ЭВМ управляет и сверлильно-фрезерным станком СМ-600Ф4 для обработки печатных плат. Станок сверлит отверстия малого диаметра в местах крепления выводов на печатной плате электронных деталей. Он может обрабатывать сразу несколько плат, собранных в четыре пакета. Это делается с высокой точностью: отклонение в координатах не превышает сотых долей миллиметра.



Специальный станок с ЧПУ (рис. 1) демонстрирует возможности вычислительной техники в автоматизации трудоемких обмоточно-изолировочных операций при изготовлении электродвигателей. Его система управления с помощью микропроцессора следит за намоткой и укладкой провода. Собранный пакет проверяется компьютером на электроизоляцию и надежность намотки провода.

В советской экспозиции демонстрировались рентгеновские вычислительные томографы, предназначенные для диагностики заболеваний головного мозга. Компьютерная томография дает врачу в 100 раз больше информации, чем традиционный рентген. Все процессы исследования мозга на томографе проводятся с помощью ЭВМ, снабженной развитым математическим обеспечением. Томограф СРТ-1010 позволяет исследовать отдельный участок мозга диаметром 220 мм, при глубине исследуемого слоя 10 мм. Специальные датчики принимают сигналы от исследуемого объекта и посылают их в ЭВМ, которая по программе обрабатывает данные (их может быть до десятка тысяч) и вычерчивает на экране дисплея изображение мозга. Врачам сразу видны в нем опухоли, травматические повреждения, которые невозможно заметить при обычной рентгеноскопии.

Мы рассказали лишь о нескольких экспонатах, представленных на «Электро-87», но и они продемонстрировали широкие возможности вычислительной техники.

В. ТАЛАНОВ
Канд. техн. наук

Международная книжная выставка-ярмарка в Москве

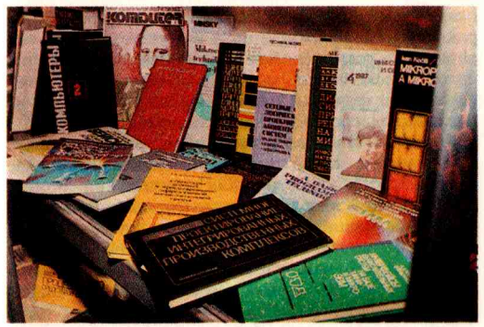
Книга — товар, но товар особого рода... потому что товар этот олицетворяет собою мысль, вдохновение, мечту...
М. О. Вольф

В сентябре на ВДНХ СССР проходила ставшая уже традиционной Московская международная книжная выставка-ярмарка. Десять лет назад в первой Московской выставке-ярмарке участвовали 1535 фирм и издательств из 67 стран и три международные организации. На шестом форуме книги свою продукцию представляли более трех тысяч фирм из 103 стран, тринадцать международных и три национальные организации. В числе участников ММКВЯ-87—ООН, ЮНЕСКО, Всемирный совет мира, Международная организация труда. Экспозиции фирм и организаций всех континентов были расположены в трех просторных павильонах ВДНХ.

ММКВЯ-87 открыла новые перспективы деловых, коммерческих связей советских книгоиздателей с их зарубежными коллегами. Но традиционные книжные выставки-ярмарки в Москве служат не только деловому сотрудничеству, международному книгообмену. Они призваны способствовать возникновению и укреплению доверия и дружбы между народами планеты. Девиз Московской международной выставки-ярмарки — «Книга на службе мира и прогресса». В нем, как отметил в приветствии участникам и гостям ММКВЯ-87 М. С. Горбачев, «заложена нравственная ответственность издателей и распространителей книги перед нынешним и грядущим поколениями».

Участие в Московской выставке-ярмарке — возможность продемонстрировать творческие способности и достижения в издательском деле. Никто, наверное, не остался равнодушным, побывав на празднике книги. Его масштабы, разнообразие тематики и оформления изданий на многих языках мира, фейерверк красок, книжная графика, репродукции, фолианты и книги-малютки — все это не может не впечатлять! Но «нельзя объять необъятное». Путешествуя среди книжных гималаев, остановимся у отдельных стендов и изданий.

На каждой выставке-ярмарке организуется специальная международная экспозиция. Тема такой экспозиции на ММКВЯ-87 — **«Книга и технический прогресс».** Последние Московские выставки-ярмарки показали, как возрос процент изданий научно-технической книги во всем мире. Книга по-прежнему остается основным хранителем и распространителем информации, ареной соревнования идей в науке. Цель экспозиции «Книга и технический прогресс» — ознакомить издателей и книгораспространителей, специалистов в разных областях науки и техники, широкий круг посетителей



выставки с литературой, отражающей успехи разных стран в развитии ведущих отраслей промышленности и основных направлений технического прогресса. В экспозиции, включавшей около 1000 изданий, приняли участие международные организации (ООН, СЭВ, ЮНЕСКО, ВОЗ), социалистические страны, Индия, Португалия, Бразилия, Аргентина, крупные английские, американские и западногерманские книгоиздательские фирмы.

Раздел экспозиции «Электроника и вычислительная техника» познакомил с новейшими достижениями зарубежных и советских специалистов. Среди книг раздела — «Проектирование операционных систем для малых ЭВМ» С. Кейслера, «Микропроцессоры», «Компьютеры» под редакцией К. Хелмса, выпущенные издательством «Мир» книги по языкам программирования, проектированию структуры баз данных японских, болгарских, советских и других специалистов.

После завершения Московской ярмарки зарубежная часть экспозиции демонстрировалась в крупных промышленных городах Сибири.

Многообразие книг, брошюр, альбомов, плакатов, открыток, представленных центральными издательствами, союзными республиками нашей страны, продемонстрировало возросший уровень оформления советских изданий, их полиграфического исполнения, а главное, богатство тематики нашей издательской продукции.

«Высшая школа» — крупнейшее в СССР издательство учебной, методической и справочной литературы для вузов, техникумов и профессионально-технических училищ — выпускает серии учебных пособий по новым направлениям науки и техники. На стендах издательства привлекают внимание серии учебных пособий для студентов технологических вузов: «Микропроцессоры» под общей редакцией Л. Н. Преснухина, «Робототехника и гибкие автоматизированные производства» под редакцией И. М. Макарова, «Системы автоматизированного проектирования». Пособия серии «Электронные вычислительные машины» предназначены для учащихся высших и средних специальных учебных заведений по специальностям гуманитарного, естественного и технического профиля. Книги серии могут быть использованы в учебном процессе профессионально-технических училищ и старших классов средней школы. Цель учебных пособий серии «Промышленные роботы», адресованных учащимся системы профтехобразования,— познакомить с новейшими достижениями отечественного и мирового станкостроения в области создания и применения промышленных роботов и робототехнических комплексов. Серия будет полезна преподавателям и мастерам производственного обучения.

В. А. Сухомлинский утверждал, что «школа — это прежде всего книга». Представительны экспозиции педагогических издательств нашей страны. Духовное богатство на стендах — для школы, для учителей, для детей, для родителей.

Особое место в выпуске литературы издательства **«Просвещение»** занимают книги о профессиях, производстве, людях труда, о техническом творчестве. Среди этих изданий — «Начала программирования» Г. И. Григаса, «Микрокалькулятор — школьнику» И. Я. Виленкина и др., «В мире роботов» А. Ю. Русецкого. В этом году издательство выпустило для учителей книгу Г. А. Звенигородского «Вычислительная техника и ее применение». Особенно она будет полезна учителям, преподающим основы информатики и вычислительной техники, которые не являются специалистами в области вычислительной техники и программирования.

Широк круг читателей у специализированного издательства **«Педагогика»**, выпускающего научно-педагогическую литературу, труды виднейших представителей отечественной и зарубежной педагогической и психологической науки, словарно-энциклопедические, научно-популярные издания. Издательство стремится, чтобы научный, идейный, художественный уровень выпускаемой литературы отвечал возрастающим требованиям времени. Среди его авторов — видные ученые, писатели, журналисты, учителя. Работникам народного образования мы рекомендуем изданную в этом году своеобразную антологию передового опыта учителей и воспитателей (В. Ф. Шаталова, С. Н. Лысенковой, И. П. Волкова, Е. Н. Ильина и др.) «Педагогический поиск» (составитель И. Н. Баженова). Очень привлекательно выглядят на стенде книги серии «Ученые — школьнику». Но привлекательность эта не только внешняя. Цель серии — в доступной и занимательной форме познакомить старшеклассников с новейшими достижениями науки, научно-технического прогресса, важнейшими общественно-политическими проблемами. Постоянно расширяется как количество ежегодно выпускаемых изданий, так и сама тематика книг. В 1987 г. в этой серии вышла книга А. А. Самарского и А. П. Михайлова «Компьютеры и жизнь».

Среди книжной продукции латвийского издательства **«Звайгзне»** значительное место занимает

литература для детей и юношества. Наше внимание привлекла книга Б. Я. Мартузана «Как Петя Бейсиков Тоню Соображалкину программировать учил», адресованная учащимся среднего и старшего школьного возраста. На республиканском конкурсе научно-популярной литературы эта книга была удостоена диплома первой степени. Проголосовали за нее и читатели: выпущенная в 1986 г. на латышском языке тиражом 30 тысяч экземпляров она быстро исчезла с прилавков. Вскоре издательство выпустило книгу на русском языке.

Книги советских издательств, представленные на Московской выставке-ярмарке, помогут школьникам, учащимся СПТУ и техникумов приобщиться к бурно развивающемуся миру электроники, кибернетики и информатики.

В экспозиции Народной Республики Болгарии мы остановились у стенда с книгами издательства «Техника». Издательская серия «Микрокомпьютерная техника для всех» предназначена для самого широкого круга читателей. Ее задача — помочь овладеть основами компьютерной грамотности. Тираж этих популярных в Болгарии изданий, почти 100 тысяч экземпляров, очень значителен для масштабов страны. Внимание многих посетителей экспозиции привлекают учебники, играющие немаловажную роль в осуществлении эксперимента, проводимого в единой средней политехнической школе НРБ. Содержание этих учебников, форма подачи материала основаны на смелых педагогических и методических идеях.

Говоря о Московской выставке-ярмарке, нельзя не сказать о такой форме развивающегося в ее рамках сотрудничества, как соглашения о совместных изданиях. Представители советских научных учреждений и фирма «Шпрингер-Ферлаг» (ФРГ) подписали договор о совместном выпуске 60-томной «Математической энциклопедии», которая будет крупнейшей в мире отраслевой энциклопедией.

Без книги была бы немыслима та вознесенная над материальным миром форма проявления жизни, которую мы именуем культурой.

Книжные выставки-ярмарки в Москве — праздник, возможность порадоваться успехам, но и серьезный экзамен, повод поразмышлять о сегодняшних проблемах, завтрашних перспективах.

Н. АЛЕКСЕЕВА

Смотреть и видеть

Заметки с выставки

В июне нынешнего года на ВДНХ СССР проводилась выставка «Информатика в жизни США», организованная американским информационным агентством ЮСИА совместно с Торгово-промышленной палатой СССР. После Москвы она отправилась в другие города страны.

В прессе уже дана оценка этой выставке, в адрес ее организаторов высказывались справедливые замечания. Нет смысла повторять их; попробуем взглянуть на экспозицию конструктивно. Что нового смогут узнать посетившие ее учителя информатики и старшеклассники?

Основная мысль выставки довольно серьезна: мы переживаем эпоху превращения индустриального общества в информационное. Правда, большинство специалистов были разочарованы: «все это уже видели», «ничего выдающегося»; удовольствие получали, как правило, от того, что удавалось «утереть хозяевам нос» — ошеломить стендиста вопросиком об операционных системах, да еще устами школьника, дескать: «знай наших». Мы во многом утратили способность беспристрастно наблюдать жизнь соседа, вслушиваться в его голос, отыскивать рациональное зерно. Нужно, чтобы не только профессионалы, но и простые посетители всерьез задумались над феноменом современной информатики, чтобы важнейшие вопросы технической политики государств не решались за спиной общественности.

Обилие информационных потоков, хорошо продемонстрированное на выставке, не только оглушает, но и заставляет задуматься. Это электронное изобилие требует от человека новых способностей: отобрать в океане сведений нужные, незамедлительно отбросить «пустые»; нужны твердые навыки ориентации в информационной среде. Объясняется оно специфическим свойством электронной информации: легкостью размножения и практической «вечностью» существования. Со всей остротой встает проблема уничтожения ненужной и неверной информации. Сегодня необходимо уметь выделить в сообщении главное, правильно сформулировать свою информационную потребность, оценить соответствие запроса и ответа, отдавать себе отчет в возможных погрешностях электронных операций.

Компьютеров как таковых на выставке мало, и все они незаметны среди другой аппаратуры. Посетители почти не задерживают на них свое внимание. Это одно из основных впечатлений: исчезновение компьютера как самостоятельной ценности. Он как бы утапливается в глубь окружающих нас предметов, растворяется в мире привычных вещей. Он встроен в кассовый аппарат, медицинские приборы, радиоаппаратуру, музыкальные инструменты, средства связи. Интересно отметить, что взаимодействие человека с такими предметами и инструментами сохраняет сравнительно традиционный характер: работа кассира или врача с новым инструментом не очень меняется и требует все меньше технических знаний.

Особого внимания заслуживает заключительный экспонат выставки — электронный опросник, сочетающий в себе видеосистему с диском, компьютер с базой данных и сенсорный экран. Даже для специалистов такой фокус — новинка. На экране телевизора появляется диктор, который вежливо предлагает ответить на ряд вопросов. Казалось бы, обычный видеофильм — многие дети видят подобные вещи у своих родителей и друзей. Но затем на экране появляется несколько квадратиков — нужно коснуться пальцем одного из них. Квадратики содержат в себе варианты ответов: «да», «нет», «не знаю» («затрудняюсь ответить»). Прикосновение пальца к одному из квадратов регистрируется системой как ответ посетителя, и в зависимости от него диалог продолжается по одной из ветвей: снова на экране симпатичный диктор что-то говорит и предлагает новые вопросы. Задумайтесь, к примеру: где хранится изображение? Система должна быстро найти нужный фрагмент видеозаписи,— значит, видеомагнитофон для нее не подходит, а нужен лазерный диск с прямым доступом.

Логика вопросов и ответов построена на основе трех вариантов ответа, в отличие от привычного в школьном курсе информатики ветвления по условию «да», «нет». Третий вариант «не могу сказать» дополняет логику диалога до более привычного мышления и обыденного общения. Экспонат таит и некоторые сюрпризы: ход диалога разнится от случая к случаю. Можно заметить, что система отмечает интенсивность ответов посетителя и при длительном интервале между вопросом и ответом теряет «интерес» к посетителю и прощается с ним. Если же у экрана оказывается активный посетитель, который дает непротиворечивые ответы, то система начинает предлагать самые сакраментальные вопросы: «Регулярно ли вы слушаете передачи «Голоса Америки»?», «Регулярно ли читаете журнал «Америка»?» и т. д. Какого рода информацию может собрать этот копьютер за время экспозиции, какие для этого нужны объемы внешней памяти, какие выводы могут быть сделаны на основе таких данных — на эти вопросы должен уметь ответить любой информационно культурный человек.

В рамках выставки проходил советско-американский семинар по проблемам информационного общества. Вел его известный американский специалист по проблемам советской информатики — профессор Сеймур Гудмен из университета штата Аризона. Он кратко представил схематическую модель информационного общества западного типа, а затем предложил присутствующим построить по аналогии советскую модель информационного общества, каким мы его себе сегодня представляем. Для западной модели профессор выделил такие признаки, как вовлечение большинства населения в информационные потоки, децентрализация управления этими потоками, ослабление государственного контроля за производством, распространением и потреблением информации. Большой интерес гостей из США вызвали проблемы развития советской информатики в новых условиях хозяйствования: коммерческая основа, конкуренция и т. д.

Лейтмотив семинара совпал с основной мыслью выставки: индустриальное общество, исчерпав свои потенциальные возможности, сменяется информационным. Разумеется, эта проблема ждет компетентного рассмотрения социологами, экономистами, специалистами других профессий.

От проблем техники разговор перешел к проблемам существования человека в копьютерном окружении. Стержневым в этом вопросе был доклад госпожи Шерри Теркл — известного американского специалиста в области психологии общения.

Конструктивным результатом семинара можно считать вывод о том, что модели информационного общества должны строиться на разных уровнях — от универсального до индивидуального. Когда информация рассматривается как ресурс, аналогичный минеральному сырью или энергии, модели ее производства, распределения и потребления могут носить интернациональный характер. При рассмотрении же взаимодействия «человек — ЭВМ» модели информационного общения должны быть достаточно разнообразными, вплоть до «личных».

Общение с американскими коллегами было интересным и полезным, хотя со многими их утверждениями, в частности об ослаблении роли централизованного контроля в информационном обществе, согласиться трудно.

В заключение — об одной тенденции, заметной сейчас в США. Там стабилизировалась численность программистов, в том числе и высококлассных («системщиков»), зато повышается спрос на специалистов в более традиционных отраслях технического прогресса: химической технологии, физике, оптике, механике и т. д. Ожидается рост и сферы обслуживания, связанной с ЭВМ (операторы, инженеры техобслуживания).

Объясняется это, вероятно, целым рядом причин. Прежде всего, программирование оказалось в последние 10—20 лет насыщено очень молодыми кадрами. Кроме того, удешевление элементной базы, появление ЭВМ новых архитектур, распространение заказных микросхем позволяют существенно поднимать эффективность систем за счет «опускания» их на уровень электроники. И наконец, немаловажна роль продемонстрированных опытом последних лет значительных сложностей при попытках содержательного проникновения программиста в традиционные предметные области; а это повышает, точнее, не дает снизить роль традиционных специалистов.

А. ДЕНИСЕНКО

Летние школы юных программистов

В Казахстане

В г. Алма-Ате с 15 по 26 июня 1987 г. проходила I республиканская летняя школа юных программистов. Открытие школы состоялось в большом зале республиканского дворца пионеров и школьников. Перед участниками выступили вице-президент АН Казахской ССР академик У. М. Султангазин, первый заместитель министра просвещения Казахской ССР А. Ф. Саламахин, партийные и советские руководители республики, представители высших учебных заведений г. Алма-Аты. Этот день завершился большим концертом, подготовленным танцевальными коллективами республиканского Дворца пионеров и школьников и Дома пионеров Калининского района.

Организаторами школы юных программистов являлись Министерство просвещения Казахской ССР, Академия наук республики, ЦК ЛКСМ Казахстана. В подготовке и проведении школы приняли участие Педагогический институт им. Абая, республиканский Дворец пионеров и школьников, Центральный институт усовершенствования учителей, Алма-Атинский областной институт усовершенствования учителей, республиканская физико-математическая школа-интернат.

Летняя школа имела в своем распоряжении 42 японские вычислительные машины «Ямаха» и 24 машины отечественного производства «Агат». Такой представительный парк электронно-вычислительной техники позволил обеспечить каждого участника 3—5 часами машинного времени ежедневно.

При регистрации 197 школьников из всех девятнадцати областей республики (из них более половины учащиеся сельских школ) были условно разделены на три потока. В первый поток вошли школьники, которые не изучали основы информатики и вычислительной техники, во второй — изучавшие безмашинный вариант курса, в третий — учащиеся, имеющие навык работы на персональных компьютерах.

Учебные планы для каждого потока, предусматривающие степень подготовленности учащихся, были разработаны отделом методики преподавания информатики и вычислительной техники НИИ ПН им. Алтынсарина под руководством кандидата физико-математических наук Ж. А. Караева. Так, учащиеся первого потока знакомились с языком программирования Бейсик, учились составлять простейшие программы для решения задач из курса математики, физики. Второй поток изучал машинный язык Бейсик более углубленно. Некоторые школьники занимались разработкой несложных игровых программ. Учащиеся третьего потока прослушали курс лекций по ассемблеру «Ямахи», графическим возможностям персональных ЭВМ. Для руководителей команд был организован семинар. Сотрудники НИИ ИВТ АПН СССР А. М. Морева и Т. И. Тихонова прочитали цикл лекций, а сотрудник Ленинградского института авиационного приборостроения Н. Н. Бровин поделился опытом преподавания предмета «Основы информатики и вычислительной техники» в IX—X классах одной из ленинградских школ. На научной конференции юных программистов учащиеся защищали свои рефераты, разработки и готовые программы. Во время работы летней школы проводилась олимпиада.

Слева — обладатель первого приза за лучший пакет программ Женя Новиков (г. Рудный, школа № 4, VIII класс)

Таня Сотниченко — второй призер конкурса на лучший пакет программ (Целиноградская область, Малиновская школа, VIII класс)

Участники летней школы посетили вычислительные центры институтов АН Казахской ССР, где для них были организованы встречи с ведущими специалистами и учеными Казахстана. Ребята побывали на экскурсиях в урочище Медео, познакомились с высокогорным спортивным комплексом, посетили ВДНХ Казахской ССР.

Надеемся, что проведение таких школ в Казахстане станет традицией.

И. ЛЫСЕНКО

В Новосибирске

Летние школы юных программистов — уже традиция в г. Новосибирске. В августе 1987 года проведена очередная — двенадцатая. Школа стала региональной (переход от статуса «всесоюзной» к статусу «региональной» отражает возросший интерес к программированию и в связи с этим — потребность в увеличении числа летних школ юных программистов). Исключение сделано для делегаций Ленинграда, Кишинева и Грузии, представители которых активно участвовали в создании и внедрении различных разработок. В школе приняли участие около 200 учащихся и преподавателей, преимущественно из Сибири и Средней Азии, делегации НРБ, ПНР и ГДР.

Техническое оснащение обеспечил НИИ ИиВТ АПН СССР, предоставивший школе два учебных кабинета (Томсон и Ямаха). Занятия вели сотрудники СО АН СССР и студенты НГУ, НЭТИ и НГПИ, а также преподаватели из Болгарии, Польши, Алма-Аты и Одессы.

В один из первых учебных дней школу юных программистов посетил академик А. П. Ершов. Он прочел лекцию и провел беседу с руководителями делегаций.

По традиции школу открыла конференция, на которой были представлены разработки программ (домашние работы учащихся — 39 сообщений). В дальнейшем на конференциях ребята отчитывались о результатах работы по решению задач, предложенных им на занятиях.

Учащиеся делились на два потока: первый изучал языки программирования Рапира и Лого, второй — Паскаль и Си. Ежедневные практические занятия подкрепляли теоретический курс. Для желающих прочитаны спецкурсы по языкам Пролог, Лисп, Форт, ассемблер PDP — 11, Z80 и 6502. Проведены семинарские занятия по программированию на Прологе (К. Кирова — НРБ), разбору олимпиадных задач, отобранных для международных конкурсов в Болгарии (П. Пеев — НРБ), по технике алгоритмизации в процессе программирования (А. М. Альт — Одесса).

Самостоятельная работа учащихся, как и в прежние годы, была организована по бригадам (42 бригады по 2—5 человек). Каждая бригада выполняла общий учебный проект. Работа курировалась членами жюри, проводившими конференцию. При подведении итогов 80 работ отмечено грамотами и дипломами.

История языка Рапира, проблемы спецификации программ, проверка их правильности и оценка качества, использование математических моделей в программировании, система «МРАМОР», разработанная в ВЦ СО АН СССР для типографии «Правда» — темы некоторых лекций. Профессор С. Валигурски (ПНР) рассказал о новых геометрических задачах, решаемых с помощью машинной графики в терминах языка Лого и показал их на ПЭВМ. Беседы о программировании (А. А. Берс) и математике (А. М. Альт), «физические вечера» (М. Ю. Брук) проведены для наиболее любознательных учащихся. Профессор Дж. Маккарти (США) прочел лекцию «Проблемы и достижения искусственного интеллекта».

На преподавательском семинаре обсуждали отечественный и международный опыт, делились замыслами, учебным программным обеспечением. Н. А. Юнерман рассказала о результатах эксперимента, проводимого в школе юных программистов и в средней школе № 166 г. Новосибирска, по преподаванию программирования учащимся разных возрастов, начиная со II класса.

Интересный пакет программ — «Исполнители», «Машинист», «Том Сойер», «Чертежник», «Робот»,— предназначенных для обучения программированию учащихся младших классов, показали школьники из г. Красноярска А. Шабанов, Д. Сухов, Б. Рисенецкий, В. Филимонов. Содержательные обучающие программы продемонстрировали школьники А. Морозов, М. Ковалев (г. Свердловск), Г. Северьянова, Е. Баранова (г. Красноярск). Можно отметить прикладные программы школьников из ГДР, г. Новосибирска, Омска.

Л. ГОРОДНЯЯ,
В. МИЛЮТИН

Всесоюзный семинар

С 6 по 11 июля 1987 г. в Минске на базе РИУУ и ГИУУ МП БССР проводился семинар, организованный Минпросом СССР совместно с АПН СССР, по обучению эксплуатации Программно-методического комплекса (ПМК) «Информатика-87» на ПЭВМ «Агат» (разработчики — сотрудники НИИ ШОТСО АПН СССР, НИИ ВК, ВЦ СО АН СССР).

ПМК «Информатика-87» предусматривает проведение 29 ч практических занятий и 7 ч лекционных. Он включает в себя методическое пособие для преподавателя по проведению лекций, методическое пособие для слушателей по проведению самостоятельных практических работ и программное обеспечение самостоятельных работ, состоящее из 33 программ с соответствующей документацией.

В содержание семинара входили лекции и практические занятия на ПЭВМ «Агат».

Цели и задачи «Программы подготовки работников народного образования по ОИВТ», их реализацию в ПМК, характеристику состава пакета и методических материалов ПМК осветила в своем выступлении старший научный сотрудник НИИ ШОТСО АПН СССР И. В. Роберт.

О программно-технических и системных возможностях ПЭВМ «Агат», о перспективах развития элементной базы и программного обеспечения этой машины, об увеличении надежности работы новой версии «Агата» рассказал старший научный сотрудник НИИ ВК А. Ю. Кривцов.

На особенностях системы «Школьница» и ее дальнейшем развитии остановился в своем выступлении сотрудник НИИ ИВТ АПН СССР В. А. Цикоза.

Опыту преподавания курса «Основы информатики и вычислительной техники» в СШ № 502 Москвы, оснащенной ПЭВМ «Агат», было посвящено выступление сотрудника НИИ школ РСФСР В. А. Самолысова.

Представители МП БССР — начальник отдела информатики и ЭВТ А. М. Терляев и инспектор школ Э. А. Лихтарович ознакомили участников семинара с работой органов народного образования БССР по внедрению компьютерной техники в учреждения просвещения.

На практических занятиях, проходивших в течение четырех дней по 4—5 часов, участники семинара имели возможность проверить работоспособность полученного пакета ПМК, состоящего из пяти дисков, ведь «Агат» — машина, не отличающаяся большой надежностью, потому так важно было не только растиражировать пакет для участников семинара, но и опробовать его на месте.

Когда информация о Всесоюзном семинаре готовилась к печати, мы получили письмо А. Лихтарович, инспектора школ отдела информатики и электронно-вычислительной техники Министерства просвещения БССР. Она добавляет к рассказанному несколько деталей. В частности, что собрались в Минске представители одиннадцати республик; что не все проходило гладко, так как последний вариант программного обеспечения был доставлен на семинар накануне закрытия; что на семинаре «делились участники своими бедами, опытом самостоятельного ремонта клавиатуры, мечтали о лучших образцах дисплеев, обучающих программ». Спасибо за откровенный рассказ!

Дети в информационный век

С 18 по 25 мая 1987 г. в Софии состоялась международная конференция «Дети в информационный век», в которой приняли участие около 400 представителей из 44 стран.

В рамках конференции была проведена встреча главных редакторов компьютерных журналов, олимпиада по информатике среди школьников, выставка вычислительной техники.

Пленарное заседание было посвящено проблемам компьютеризированного общества. Основной доклад на открытии сделал профессор С. Пейперт (США).

В работе трех секций конференции приняли участие известные ученые: профессор Шиба (Япония), Тинслей (Великобритания), академик А. Ершов (СССР).

Журнал «Информатика и образование» опубликует наиболее интересные сообщения. В этом номере мы предлагаем вниманию читателей статью А. Ершова, в основу которой положен доклад, сделанный на конференции «Дети в информационный век».

Если сможете — купите

Пожалуй, эта книга* — для школьников, но это не учебник. Ее стиль напоминает стиль лучших книг Я. Перельмана — нечто вроде сборника рассказов, порой новелл, посвященных разнообразным аспектам применения компьютеров в производстве и на транспорте, в торговле и криминалистике, медицине и учебном процессе.

Впрочем, эта книга и для взрослых. Проблемы компьютеризации в ней рассматриваются не только широко и популярно, но и глубоко и серьезно. Даже программист найдет что-нибудь интересное для себя, и уж конечно — учитель информатики.

Авторы хорошо знают тему и, чувствуется, пристрастны к вычислительной технике — порой это подводит их. Да, ЭВМ поможет наладить учет и предупредить некоторые злоупотребления в торговле, но принципиально исключить случаи воровства и обмана никакая техника не способна.

Курс ОИВТ «отнял» часы у литературы, однако ознакомление с этой книгой поможет убедить преподавателей этого предмета в обоснованности такого «ущемления». Кроме того, информацию из нее можно использовать в работе университетов педагогических знаний, в частности для бесед с родителями старшеклассников, чтобы помочь им понять, чем занимаются дети на уроках по новому предмету.

Книга быстро исчезает с прилавков магазинов, и можно только сожалеть, что она напечатана столь небольшим — 50 000 экз.— тиражом.

В. КРОНГАУЗ,
директор СШ № 345 г. Москвы
Г. БАЛДИН,
учитель информатики СШ № 345 г. Москвы
А. ВОЛОДАРСКИЙ,
с. н. с. ИИЕТ АН СССР

* *Халамайзер А. Я., Перегудов М. А.* Что может компьютер? М.: Московский рабочий, 1987.

ЧТО МОЖЕТ ЭВМ?

...И спасти от разорения

После 1981 г., когда концерн «Крайслер» был на грани банкротства, его дела значительно улучшились — в первую очередь благодаря повышению производительности и гибкости производства, достигнутых более широким внедрением компьютеров.

ЭВМ используются в системах автоматизированного проектирования и автоматизации инженерного труда, массовое внедрение которых дало трехкратный рост производительности. Роботы изготовляют обивку для автомобилей, декоративные деревянные панели практически без участия людей. Сейчас фирма пытается «замкнуть» цепь «автоматизированное проектирование — автоматизированное изготовление комплектующих деталей — автоматизированная сборка». Специалисты «Крайслера» считают, что ключевым элементом этого комплекса должна стать единая база данных.

Изобретательская пассивность

Научно-производственное объединение «Поиск» ежегодно выпускает предметно-статистический указатель «Тематические направления изобретательской активности в СССР и промышленно развитых капиталистических странах». К нашему журналу он имеет тройное отношение: во-первых, как пример практического использования информатики — ведь этот указатель на основе огромного количества информации о патентах и авторских свидетельств дает тематический «срез», на котором отчетливо видны мировые тенденции изобретательской деятельности, порой весьма полезные для коррекции деятельности советских ученых.

Во-вторых, данные для анализа готовятся с помощью ЭВМ, что, в частности, делает их более наглядными.

Ну и, в-третьих, сами данные. По динамическим устройствам памяти для ЭВМ удельный вес изобретений (в рамках отрасли) в СССР — 2 %, в Японии — 14 %; по полупроводниковым приборам в СССР — 2 %, в Японии — 21 %. Пассивны наши изобретатели в этих областях...

Люди важнее

Значительные успехи японских промышленных корпораций на рынках Америки застали американских предпринимателей врасплох. Придя в себя, они бросились вдогонку за японцами в самом важном, как им казалось, аспекте производства — в уровне автоматизации.

Наибольшую активность проявила «Дженерал моторс», вложившая за последние 7 лет в модернизацию своих заводов 40 млрд. долларов. Однако даже на самом своем современном предприятии эта фирма не смогла превзойти результатов, достигнутых на американском по положению и японском по организации труда заводе «Нью юнайтед мотор мэньюфекчуринг», построенном в Калифорнии совместно с «Тойотой». Секрет оказался в умелом планировании производственного процесса, а не в дорогостоящих роботах.

В заключение — небольшая цитата из «Файненшл таймс».

«Некоторые специалисты, ранее призывавшие противопоставить людям роботы, теперь меняют взгляды и говорят, что автоматизация — лишь один из многих путей к увеличению конкурентоспособности».

Напечатано в 1987 г.

К сведению читателей

Международный научно-исследовательский институт проблем управления и научно-производственное объединение «Центрпрограммсистем» начинают с 1988 г. выпуск приложения к международному журналу *«Проблемы теории и практики управления»*.

Приложение *«Программные продукты и системы»* (индекс по каталогу Союзпечати 70754) распространяется по подписке, а также поступает в розничную продажу. Подписка с любого квартала года. Стоимость одного номера — 2 рубля. Периодичность выпуска — 4 раза в год. Приложение рассчитано на широкий круг читателей и освещает практические вопросы использования программ в самых различных областях деятельности.

Обычно это оказывается школа. Школы я люблю, работать в них очень интересно, хотя и трудно: молодежь бывает не только настырна, но и небрежна. Однако все искупается сознанием своей нужности.

Ведь я хоть и маленькая, но ЭВМ и могу делать все, что могли мои

старшие родственники, которые занимают большие залы и лет 15 назад были новинками техники. Тот, кто сумеет освоить все мои возможности, будет уверенно чувствовать себя при встрече с любым компьютером. А вечера приятнее всего проводить дома, с детьми...

ИНФО

6 '87

ИНФОРМАТИКА И ОБРАЗОВАНИЕ

Информатика и образование, 1987, № 6, 1—128