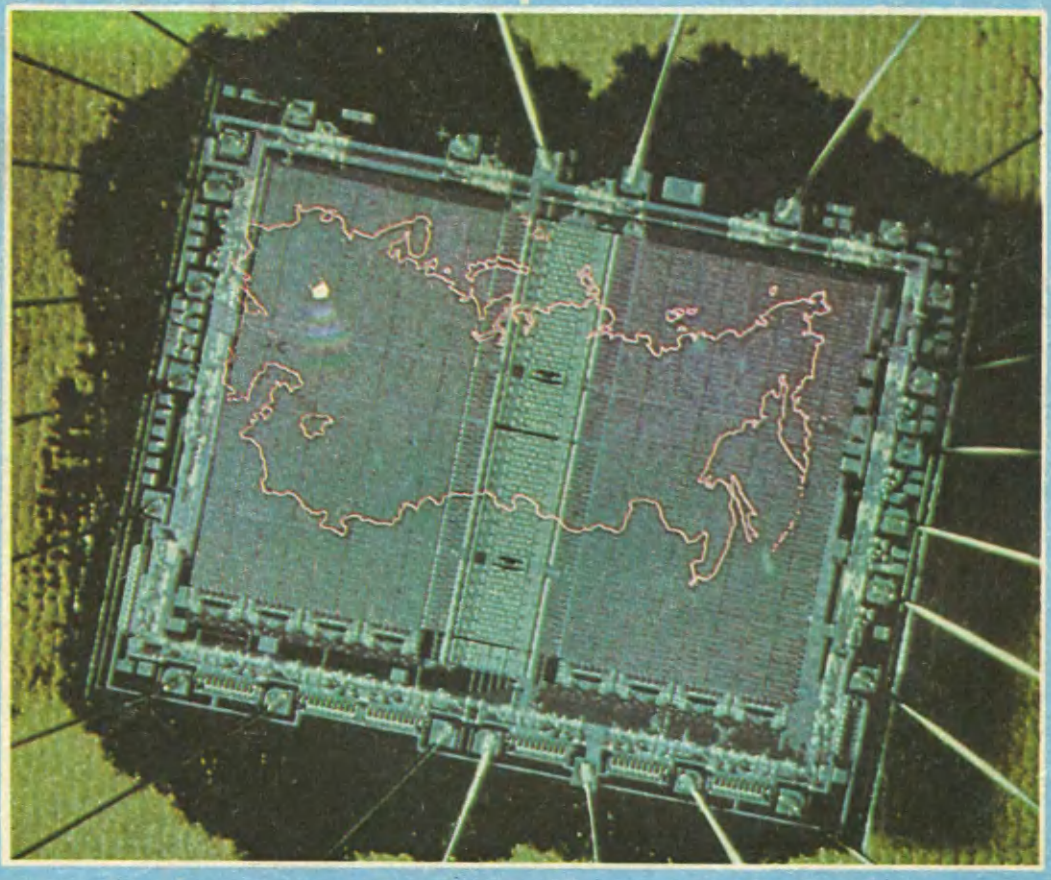
ISSN 0234—0453

ИНФО

ИНФОРМАТИКА И ОБРАЗОВАНИЕ

5 1987



Путь советской школы за 70 лет

(2-я и 3-я страницы 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

ИНФОРМАТИКА И ОБРАЗОВАНИЕ

5 '87
сентябрь — октябрь

Содержание

Внеклассная работа

Молодежная инициатива

Точка зрения

Нам пишут

Информация

Веселый урок

Обложка Э. Бажилина, М. Вылегжанина

В оформлении номера принимали участие Э. Бажилин, М. Вылегжанин, А. Лобашинский, А. Пономарев, С. Расторгуев, В. Шкрабан

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 27.07.87. Подписано в печать 02.09.87. А 07409. Формат 70×100/16. Печать офсетная. Усл. печ. л. 10,40. Уч.-изд. л. 14,38. Усл. кр.-отт. 42,88. Тираж 90780 экз. Заказ 2173. Цена 60 коп.

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

У истоков

Наша страна отмечает 70-летие Октября, переживая захватывающий период в своей истории — период глубокого обновления и коренной модернизации всех сторон жизни общества. Важнейшая роль в перестройке экономики, в ускорении научно-технического прогресса принадлежит вычислительной технике.

В ноябре этого года советская наука отмечает знаменательный юбилей — 85 лет со дня рождения Сергея Алексеевича Лебедева, крупного ученого и организатора науки, основоположника отечественной вычислительной техники.

Год 1947... Феофания. Здесь, под Киевом, при Институте электротехники АН УССР создана лаборатория, в которой суждено было родиться первой советской электронно-вычислительной машине, названной МЭСМ (малая электронная счетная машина). Позднее, лаборатория, сыгравшая столь значительную роль в становлении отечественной вычислительной техники, была преобразована в Институт кибернетики. Возле здания института установлен памятник МЭСМ, памятник творческому коллективу создателей машины, сделавшему, несмотря на трудности послевоенных лет, первые шаги в развитии электронной техники.

Возглавлявший институт электротехники с 1946 года молодой академик АН УССР Сергей Алексеевич Лебедев, в свои 45 лет уже известный ученый, проявив научную принципиальность и гражданское мужество, выбрал новое прогрессивное направление в науке. Он понимал, что для успешного решения важнейших задач народного хозяйства и обороны страны необходимы более совершенные вычислительные машины, быстродействие которым обеспечит электроника, автоматизацию — программное управление.

Научная и инженерная деятельность С. А. Лебедева (1902—1974) началась в период осуществления плана ГОЭЛРО. После окончания электротехнического факультета МВТУ им. Н. Э. Баумана он занимался проблемой устойчивости энергосистем, важной для решения задач объединения крупных электростанций для дальних электропередач. Уже в те

годы он убедился в необходимости быстрого моделирования сложных систем и решения трудоемких задач.

В 1943 году Сергей Алексеевич вступил в ряды Коммунистической партии Советского Союза, работая для обороны Родины с первых дней Великой Отечественной войны. 1953 год — избрание Лебедева действительным членом АН СССР. С этого года в течение двадцати лет руководил он Институтом точной механики и вычислительной техники АН СССР, носящим теперь его имя. Важнейший этап в развитии отечественной вычислительной техники — создание первой советской быстродействующей электронной счетной машины (БЭСМ-1), признанной тогда одной из лучших в мире. БЭСМ-1, построенная под руководством С. А. Лебедева, явилась прообразом вычислительных машин высокого класса.

В 1956 году Сергею Алексеевичу было присвоено звание Героя Социалистического Труда. Успех деятельности Лебедева объясняется не только его талантом, опытом и глубокими знаниями ученого-исследователя, конструктора, инженера. Этот человек был яркой, незаурядной личностью, обладал высокими нравственными качествами, огромной работоспособностью. Он умел отстаивать свои научные взгляды, свою жизненную позицию. Понимая, что разработка ЭВМ — дело коллективное, умел организовывать творческие коллективы и эффективно руководил ими, поддерживая атмосферу взаимопонимания и воодушевления.

Лебедев доверял молодым, верил в них, привлекал к работе над серьезными проектами, считая, что лучшая школа для специалиста — участие в конкретных разработках. Большое значение он придавал подготовке научных кадров, являлся основателем и руководителем кафедры вычислительной техники Московского физико-технического института.

Создание в 60-х годах полупроводниковых машин, в частности создание под руководством С. А. Лебедева БЭСМ-6,— новый этап в развитии вычислительной техники. Разработка БЭСМ-6, по своей элементной базе машины второго поколения, классический пример проектирования перспективных ЭВМ. В ней были заложены решения, широко использующиеся при создании машин третьего и четвертого поколения. БЭСМ-6 и сейчас успешно работают в крупных вычислительных центрах страны.

Исключительный талант Лебедева проявлялся и в научном предвидении. Ставя цель, он видел перспективы развития структуры будущей машины, правильно выбирал средства ее реализации, учитывая возможности отечественной электронной промышленности.

Велика роль С. А. Лебедева в разработке математического обеспечения — одним из первых он понял значение совместной работы математиков и инженеров при создании вычислительных систем. Кроме фундаментальных разработок С. А. Лебедев участвовал в создании многомашинных и многопроцессорных комплексов, операционных систем, систем программирования.

Авторитетна созданная С. А. Лебедевым научная школа. Сейчас по всей стране в намеченных им главных направлениях развития электронно-вычислительной техники работают его ученики, целые научные коллективы. В честь юбилея ученого проводят конференции, Лебедевские чтения.

Дорогой долгой и трудной, преодолевая противоречия, взаимно адаптируясь, но непременно совершенствуясь, идут рядом человек и машина. Совершенствуя технику, люди совершенствуют себя. Современная вычислительная техника — катализатор научно-технического прогресса, эффективное средство расширения наших возможностей. В Советском Союзе у истоков ее развития стоял выдающийся ученый, замечательный человек — Сергей Алексеевич Лебедев.

В. ТОЛСТОШЕЕВ

Компьютеризация и правовая подготовка специалиста

На июньском (1987 г.) Пленуме ЦК КПСС подчеркнута важность высокой организованности и сознательной дисциплины, строгого соблюдения порядка и правил жизни нашего общества, развития активности, инициативы в рамках закона. Это предъявляет повышенные требования к правовой подготовке специалистов с учетом использования ими ЭВМ.

Компьютеры все шире используются в работе правоохранительных органов, с помощью ЭВМ в стране организуется централизованный учет законодательных актов и ведомственных общеобязательных документов, в отраслях вводится автоматизированный контроль за соблюдением предприятиями договорной дисциплины, в аппаратах министерств и ведомств создаются автоматизированные системы по законодательству.

Электронно-вычислительная техника берется на вооружение при подготовке различных официальных документов. (Например, информационно-вычислительный центр ЦК ВЛКСМ на своей базе осуществил учет и анализ предложений во время обсуждения проекта Устава комсомола. Специальной комиссии ИВЦ регулярно представлял распечатки с характеристикой поступающих предложений, а последующие решения об их принятии или отклонении снова вводились в память ЭВМ).

Сейчас ведется разработка проектов автоматизированных рабочих мест на базе персональных ЭВМ для специалистов в области права. В перспективе возможны более высокий уровень автоматизации юридической деятельности, создание своеобразного искусственного «юридического интеллекта» в виде электронных кодексов, баз правовых знаний, автоматизированных экспертно-правовых систем.

Связь между компьютерами и правовым образованием не односторонняя: овладение соответствующими правовыми знаниями приобретает все большее значение для рационального применения ЭВМ в экономике, культуре и других сферах. Ведь правильное использование ЭВМ в них невозможно без знаний участниками отношений, возникающих в процессе компьютеризации, основ правовой регламентации.

Знакомство с достижениями правовой науки может быть полезно при изучении некоторых понятий информатики, в частности алгоритма. Такие его свойства, как точность, последовательность, адресность и другие, давно используются при характеристике правовых явлений.

В стране интенсивно развиваются производство, эксплуатация и обслуживание электронной техники. Создаются разнообразные межвузовские вычислительные центры, сети коллективного пользования, центры программирования, единый фонд алгоритмов и программ высшей школы, общесоюзный банк информации для компьютерного обучения.

Чтобы получить нужную информацию из центральных ЭВМ, необходимо знать порядок доступа к ним и распоряжения ими. Обращение к услугам специализированных организаций, обслуживающих ЭВМ, неизбежно связано с выяснением правовых условий (установленных цен, гарантий и т. д.) пользования их сервисом. Без определенных юридических знаний не обойтись при решении конкретных задач с использованием

электронной техники. Такая потребность может возникнуть уже с первых шагов трудовой деятельности, в частности при заключении трудового договора или оформлении трудовой книжки с помощью ЭВМ.

Для работы на автоматизированном рабочем месте в ряде случаев нужно иметь представление о правовом режиме доступа к нормативным массивам, использования служебной информации и передачи ее абонентам. Многим служащим, повседневно сталкивающимся с выполнением формальных процедур, в условиях широкой автоматизации труда необходимы будут также специальные юридические знания.

При желании непосредственно участвовать в жизни своего трудового коллектива с помощью персональных компьютеров при временной нетрудоспособности, во время отпуска или отсутствия на работе по другим причинам потребуются знания правовых форм такого подключения. В процессе получения услуг предприятий связи, сберегательных касс и иных сервисных служб с помощью ЭВМ в быту нужно иметь представление о юридических аспектах совершения конкретных действий и их последствий.

Интеграция информатики и юриспруденции — закономерный процесс в развитии гуманитарных и естественнонаучных сфер, сопровождающийся появлением новых знаний. Так, поиск законодательной информации в памяти ЭВМ требует не только технических познаний, но и юридических, касающихся структуры норм права, условий их действий и т. п. В ходе компьютеризации объем и значение таких синтетических знаний будет возрастать.

Совокупность специальных знаний об использовании электронной техники в юриспруденции и о правовых аспектах компьютеризации других сфер можно обозначить как особую область — правовую информатику. Знания в этой области имеют прикладной характер и связаны с комплексным использованием идей информатики и правоведения. По ряду объективных причин пока потребность в их изучении не столь ощутима. Отечественная промышленность только приступает к массовому изготовлению персональных компьютеров, а именно с широким применением последних резко возрастает потребность в таких знаниях.

Но не следует ждать, когда недостаток в таких знаниях станет тормозом продвижения вперед. Практика уже не раз подтверждала бесперспективность построения обучения исходя лишь из текущих потребностей без учета тенденций общественного развития. Поэтому сегодня следует позаботиться о вооружении соответствующими знаниями тех, кому предстоит трудиться и жить в иных условиях. Ведь уже к концу 1990 г. в стране должно быть изготовлено более миллиона персональных компьютеров, внедрена многотысячная сеть автоматизированных рабочих мест.

Нынешняя организация компьютерного всеобуча и правовой подготовки пока не обеспечивает их интеграции и комплексного использования. Хотя право и информатику начинают изучать уже в старших классах, должных междисциплинарных связей между ними нет. В вузах изучение использования ЭВМ по специальной дисциплине — правовой кибернетике — предусмотрено только для будущих юристов. Да и то из всех учебных заведений до последнего времени лишь в МГУ она имела обязательный характер, а в остальных факультативный. Для выпускников неюридического профиля правовой курс не предусматривает изучение подобных вопросов. Правда, в отдельных учебниках по советскому праву встречаются разделы по правовому обеспечению АСУ, но изложены они без должной связи с другими аспектами компьютеризации.

Для овладения такими специфическими знаниями необходима соответствующая материально-техническая база, позволяющая приобрести навыки практического применения ЭВМ. Пока ни в одном юридическом учебном заведении нет компьютерной базы, достаточных средств для привлечения специалистов и аренды ЭВМ сторонних организаций. Только в прошлом году при юридическом факультете МГУ создана лаборатория по применению ЭВМ в учебном процессе, оснащение которой затягивается.

Электронная техника в неюридических учебных заведениях, нередко загруженная не на полную мощность, почти не используется для правовой подготовки. В процессе компьютерного обучения даже старшеклассникам предлагают решать оригинальные технические, химические и игровые задачи, к примеру «Высадка на Луну», а вот правовые аспекты применения ЭВМ в реальной действительности остаются в стороне. К нуждам общеюридического образования не приспосабливаются еще функционирующие в ряде вузов автоматизированные обучающие системы. Нередко в оправдание такого отношения к изучению права можно слышать ссылки на специфичность этого предмета и отсутствие специальных программ, в том числе для ввода изучаемых законодательных актов в память ЭВМ.

Между тем можно использовать ЭВМ в правовой подготовке, в частности, для анализа законодательных актов. Эксперимент,

проведенный сотрудниками Института проблем кибернетики АН СССР по вводу проекта Закона о государственном предприятии в базу данных, ускоренному и сканирующему поиску его норм, подтверждает, например, эффективность изучения нормативных актов с использованием ЭВМ. Он осуществлялся на программном комплексе АСПИД-7/СМ с помощью программы «Информатор» в диалоговом режиме с выдачей результатов на экран видеотерминала и бумагу.

Определенный опыт ведения текстов нормативных документов на технической базе накоплен Научным центром правовой информации ВНИИ советского законодательства. Специфические правовые задачи с помощью электронной техники решаются другими отраслевыми юридическими учреждениями. Ряд обучающих программ в диалоговом режиме, к примеру, по теме «Гражданство» уже подготовлен лабораторией применения ЭВМ в учебном процессе юрфака МГУ.

Усиление связей между компьютерным и правовым образованием требует комплексного подхода. Первоочередным шагом следует считать включение в разные стадии юридических аспектов компьютеризации и вопросов применения ЭВМ для решения правовых задач.

Уже у старшеклассников целесообразно формировать элементарные представления о правовой стороне применения компьютеров в общественной жизни, будущей трудовой деятельности, в быту, при удовлетворении культурных и иных запросов.

Знакомство с этими вопросами целесообразно предусмотреть в учебном плане по правоведению. К примеру, при рассмотрении конституционных основ государственной жизни, где освещается роль общественных организаций, можно рассказать о компьютерных клубах и центрах. В процессе изучения основ трудового и иных отраслей права следовало бы коснуться использования ЭВМ в регламентации конкретных общественных отношений. Конечно, для этого потребуется в централизованном порядке обеспечить преподавателей методическими материалами и практическими данными.

При изучении основ информатики следовало бы также знакомить учащихся с алгоритмами решения простейших правовых задач, возникающих в повседневной жизни. Эксперимент по преподаванию различных школьных предметов с помощью ЭВМ должен, на наш взгляд, охватить и право. Следует подумать о разработке специальных «электронных» кодексов, из которых можно было бы черпать знания о конституционном, административном и другом законодательстве. Применение электронной техники способствовало бы не только внедрению активных методов усвоения юридической информации учащимися, улучшению контроля за успеваемостью каждого из них, но и повышению престижа правового предмета.

Обучающимся, которым предстоит специализироваться на внедрении, эксплуатации и обслуживании компьютерной техники, разумеется, необходимы более емкие знания, чем просто пользователя ЭВМ. Первым потребуется знакомство с законодательными основами индустрии информатики, правовым положением их звеньев, с регламентацией работы в них, особенностями трудовых прав и обязанностей операторов, программистов, электронщиков и др. Определенные знания понадобятся также тем, кто непосредственно не будет занят в индустрии информатики, поскольку требования к профессиональной компетентности определяются прежде всего видом деятельности. В связи с тем, что всем специалистам, хотя и в разной степени, придется сталкиваться с применением норм права, помимо общих вопросов компьютеризации необходимо предусмотреть их ознакомление с использованием ЭВМ в правовой работе, исходя из профиля труда.

Поэтому приобретение навыков использования ЭВМ для ориентации в конкретной обстановке, в том числе при моделировании правового поведения, анализе официальных документов, поиске нужного законодательства, следует включать в отводимое обучающимся машинное время.

Совершенствовать правовую подготовку можно было бы при широком использовании имеющейся в учебных заведениях электронной техники. Наряду с приспособлением автоматизированных обучающих систем к нуждам юридического всеобуча следовало бы приступить к разработке баз правовых знаний с учетом профиля учебного заведения. К ним можно было бы обращаться не только в процессе изучения права, но и в случае возникновения у студентов конкретных жизненных вопросов, решение которых требует соответствующих юридических знаний.

Конечно, законодательные основы индустрии информатики и применение ЭВМ в правоведении в наибольшей степени должны изучать будущие юристы, которым придется не только использовать компьютеры в своей профессиональной деятельности, но и консультировать других. При этом их кругозор, не ограничиваясь практикой учреждений юстиции, должен охватывать работу предприятий и организаций, аппарата государственных и общественных органов. В профессиональной деятельности юристов

важное место займут автоматизированные экспертно-правовые системы. Они смогут выдавать ответы на поставленные вопросы и приводить положенные в их основу аргументы, уяснение которых весьма важно для юристов.

В настоящее время, учитывая тенденцию к сокращению учебного материала и загрузки обучающихся, существующий дефицит учебного времени на изучение права, возможны возражения против обучения новым знаниям. Однако современная подготовка специалистов должна вестись с опережением потребности в кадрах.

Включение специальных разделов в новые учебные планы и программы, которые предстоит подготовить в ближайшие годы, диктуется бурно развивающимся научно-техническим прогрессом. Кроме того, можно реализовать предоставленное теперь учебным заведениям право использовать по своему усмотрению до 15 % учебного времени для оперативного отражения в планах и программах новейших достижений науки и техники.

Поставленная задача всесторонней компьютеризации учебного процесса должна в полной мере охватить правовой всеобуч и подготовку юристов. Наряду с укреплением учебно-лабораторной базы юридических учебных заведений, активным использованием ЭВМ других вузов к правовой подготовке студентов следовало бы шире привлекать отраслевые юридические институты, имеющие ЭВМ, и академические коллективы, располагающие высоким научным потенциалом. Так, Научному центру правовой информации ВНИИ советского законодательства целесообразно, на наш взгляд, поручить централизованную подготовку для учебных заведений страны информацию о новом законодательстве по изучаемым отраслям, а также выполнять иные государственные заказы для средней и высшей школы. Юридические кадры академических коллективов могли бы участвовать в разработке концепции электронных кодексов, баз правовых знаний, автоматизированных экспертно-правовых систем.

В существенном повышении нуждается профессиональный уровень преподавателей, участвующих в юридическом всеобуче. Из более чем 60 тыс. преподавателей права общеобразовательной школы и около 5 % имеют специальную подготовку. Одной из форм переподготовки, видимо, могли бы стать вечерние курсы при вузах. Для граждан, желающих приобрести знания, умения и навыки в овладении ЭВМ, перестройкой образования предусмотрено создание хозрасчетных курсов.

Научно-техническое руководство учебными заведениями, пути решения возникающих проблем электронизации правового всеобуча, его интеграции с компьютерной грамотностью должны быть в центре внимания учебно-методического объединения по специальности «Правоведение». На него необходимо возложить разработку учебных планов, программ и других учебно-методических материалов, обобщение и распространение передового опыта в этой области. Это объединение могло бы заняться централизованной подготовкой программных продуктов для обучения праву, разработкой проекта автоматизированной обучающей системы в области юриспруденции.

В. РУБЦОВ, А. МАРГОЛИС, А. ПАЖИТНОВ

Компьютер как средство учебного моделирования

Важным средством развития мышления детей в процессе обучения являются творческие формы решения школьниками учебных задач. В процессе анализа, направленного на выявление содержания объектов усвоения, выполняются особые учебно-познавательные действия. Наиболее существенные среди них — такое преобразование объекта, при котором выделяется исходное отношение (см.: В. В. Давыдов, с. 127—132), характеризующее изучаемый объект как целостную систему, а также моделирование и вычленение свойств, определяющих это исходное отношение и применение соответствующей модели объекта при решении конкретно-практических задач определенного типа. Отсюда становится понятным, почему использование компьютеров как *универсальных средств моделирования* является одним из перспективных направлений совершенствования учебного процесса.

Ряд положений, определяющих специфи-

ку, место и функции учебного моделирования, основанного на использовании ЭВМ, можно сформулировать следующим образом.

1. Компьютерные модели необходимо создавать на основе содержательного анализа объектов усвоения; разному содержанию должны соответствовать разные системы учебного моделирования. При этом одна и та же система может удовлетворять требованиям различных учебных предметов.

2. Всякое учебное моделирование, в том числе и компьютерное, создается для усвоения системы понятий. Причем целостность этой системы определяется целостностью и внутренней связью моделирующих ее действий и операций. При этом работа учащихся строится на такой логике: от исходных действий и операций к их сложной совокупности.

3. Поскольку при компьютерном моделировании в первую очередь преследуется цель передачи учащимся операционального содержания понятий*, то при разработке программ, различая предметный и операциональный аспекты моделирования, необходимо учитывать ведущую роль операциональной стороны, обеспечивающей развернутый анализ содержания объекта самим учащимся.

4. Учащиеся должны активно воздействовать на среду с целью нахождения исходного отношения и контролировать свои действия, учитывая поставленные перед ними задачи.

5. Учебное моделирование органически входит в систему учебных задач и игр, являясь своеобразным конструктором, и выступает как средство организации следующих форм совместной деятельности учащихся, учителя и учащихся:

разделение и кооперация действий и операций в структуре решения задачи между разными участниками;

взаимный контроль и оценка действий учащимися при решении учебных задач в определенной последовательности;

совместное моделирование задаваемых взрослым схем преобразования объекта;

рефлексивное отображение и представление одним учащимся способа решения задачи, осуществленного другим.

6. При разработке компьютерных моделей нужно учитывать возрастной аспект развития детей. Разным возрастным периодам должны соответствовать различные формы учебного моделирования (от квазипредметных игровых форм в младшем школьном возрасте до квазиисследовательских продуктивных и творческих форм в среднем и старшем школьном возрасте).

На основе этих положений была составлена программа учебного моделирования по курсу «Электростатика» (VII—IX класс). Задачи этого курса — усвоение учащимися системы основных понятий, таких, как электрическое поле, силовая линия, характер взаимодействия, знак электрического заряда, протяженный и точечный заряд, однородное электрическое поле, а также усвоение обобщенных способов решения соответствующих электростатических задач.

Рассмотрим несколько фрагментов, характеризующих своеобразие учебного моделирования на начальных этапах освоения понятий электростатики.

Введение в предметную среду начинается с игрового задания, которое состоит в том, чтобы на основе прицеливания электростатической пушки, заряженной пробным зарядом, поразить мишень. Процедура прицеливания выполняется детьми совместно, путем задания угла поворота ствола пушки, который можно ориентировать произвольно. Пушка заряжается пробными зарядами — «снарядами», раскрашенными в различные цвета, часть которых соответствует отрицательным по знаку зарядам, часть — положительным. На экране дисплея высвечивается мишень и электростатическая пушка. А реальная модель электростатической среды, представляющей собой общее поле системы двух протяженных электрических зарядов (положительного и отрицательного), задана учащимся неявно.

Фрагмент 1. Первая позиция мишени выбирается таким образом, чтобы мишень находилась на кратчайшей линии, соединяющей протяженные заряды. Учащиеся, руководствуясь сначала только обыденными представлениями о прицеливании, могут поразить мишень в том случае, если ее позиция соответствует описанной выше. После того как данный способ прицеливания зафиксирован учащимися, они применяют его для поражения мишени, находящейся во второй позиции. Однако, не учитывая предметно-содержательных особенностей среды (наличия протяженных зарядов и силовых линий), они не могут попасть в эту мишень. Причем в том случае, когда снаряд положительный, школьник попадает в расположенный за мишенью протяженный отрицательный заряд. На экране дисплея остаются след движения снаряда и точка его попадания в протяженный заряд.

Выполнив серию пробных выстрелов, учащиеся намечают контуры протяженных источников и некоторое число силовых линий

*Мы используем понятие соотношения действия и операции, данное А. Н. Леонтьевым — см.: А. Н. Леонтьев и современная психология: Сб. статей памяти А. Н. Леонтьева. М., 1983. С. 30.

их общего поля. Кроме того, они должны произвести классификацию с возможным переобозначением пробных зарядов-«снарядов» на «хорошие», т. е. вылетающие из пушки по направлению к мишени, и те, которые летят в противоположную сторону, высвечивая при попадании контуры расположенного за пушкой положительного протяженного источника. В результате выполнения первого задания учащиеся должны понять, что изучаемая предметная среда оказалась гораздо сложнее, чем они первоначально предполагали, и что поведение снаряда в этой среде происходит по законам, которые необходимо изучить для выполнения поставленных игровых целей. Необходимо также установить зависимость характера взаимодействия и знаков зарядов. Задание считается выполненным в том случае, если учащиеся овладели обобщенным способом поражения любой мишени, лежащей на силовой линии общего поля двух протяженных источников.

Важно, чтобы уже в процессе выполнения первого задания учащиеся отметили, что при малой начальной скорости снаряд, пролетев немного в первоначальном направлении, «увлекается» полем и летит далее вдоль одной из силовых линий.

Фрагмент 2. Во втором задании мишень располагается в области слабого силового поля, и поразить ее прежним способом невозможно. Поражение мишени происходит только в случае увеличения начальной скорости снаряда. Проведя серию пробных запусков, учащиеся должны обратить внимание на особенности старта снаряда и начать поиск способа увеличения его начальной скорости.

Следует отметить, что электростатическая пушка представляет собой размещенный в стволе независимый точечный источник, с которым пробный заряд взаимодействует перед выстрелом во время прицеливания, набирая все большую начальную скорость. Если в процессе выполнения первого задания этот эффект был побочным и незначительным (так как, стремясь скорее поразить мишень, которая высвечивалась на экране в течение ограниченного времени, учащиеся быстро производили выстрел), то теперь он превращается в реальный механизм увеличения начальной скорости пробного заряда.

Выполняя задания, учащиеся осваивают общий способ управления движением заряженной частицей в поле протяженных разноименно заряженных источников. Задание можно считать успешно выполненным в случае овладения учащимися механизмом поражения любой мишени, лежащей в области слабого силового поля.

При выполнении этого задания дети совместно осваивают деятельность прицеливания и сообщения начальной скорости, моделируя объект своего изучения.

Фрагмент 3. Третье задание связано с расположением мишени в той области изучаемой предметной среды, где, действуя уже известным способом, учащиеся не могут поразить мишень. В этом задании поражение мишени достигается путем преобразования детьми самого объекта (электрического поля протяженных источников), изменения формы силовых линий поля. Это возможно осуществить, увеличив расстояние между источниками. Прежде всего учащиеся должны обратить внимание на то, что при попадании снаряда в протяженный источник последний изменяет свое положение, чем обусловлено и незначительное изменение формы силовых линий. Данный эффект должен привести к выводу о зависимости формы силовых линий от расстояния между источниками; знание этого и определяет способ преобразования объекта. Путем направленной «бомбардировки» протяженных источников пробными зарядами можно существенно изменить форму силовых линий их общего поля и, поместив мишень на позицию, когда расстояние между протяженными источниками значительно увеличится, достичь игровой цели данного задания.

Здесь мы сталкиваем учащихся с ситуацией, при которой исследуемое ранее общее поле протяженных источников распадается на два невзаимодействующих электростатических поля тех же источников.

Важно отметить совместный характер этого этапа деятельности. Он выражен в обмене действиями и их результатами между участниками игры. Результатом такой совместной деятельности является выделение существенных закономерностей (например, перпендикулярности силовой линии поверхности протяженного источника) и, что особенно важно, получение исходных содержательных абстракций, таких, например, как точечный заряд и однородное электростатическое поле.

Средства преобразования протяженных источников представлены в виде побочных, первоначально несущественных эффектов уже в первом задании. Попадание пробного заряда в протяженный источник вызывает незначительное изменение его формы (в соответствии с законом сохранения энергии). Только с появлением необходимости в этом эффекте он становится значимым, и требуется специальный поиск учащимися средств его усиления, заложенных в изменении способов действия.

Выполнение предыдущей серии игровых

заданий позволяет учащимся перейти к изучению системы точечных зарядов, а также однородного электростатического поля (второй полученной ими содержательной абстракции). Этот этап моделирования приводит к изучению основных качественных закономерностей полей точечных зарядов, принципа суперпозиции полей, а также количественных закономерностей, выраженных в законе Кулона. Проделав эту часть квазиисследования, обнаружив и зафиксировав все основные закономерности квазиэлектростатической среды, учащиеся осваивают обобщенный способ ее преобразования, что, в свою очередь, открывает возможность для синтеза полученных абстракций в процессе реконструкции любых полей, вплоть до общего поля протяженных источников (иного, чем исходное).

Важнейшее требование к разработке компьютерных моделей — исходно задаваемая полнота предметного содержания изучаемого объекта и полная система его возможных преобразований. Так, уже в первом задании содержание квазиэлектростатической среды задается полностью, хотя и неявно для учащихся. Этим обусловлены этапы ее моделирования и освоения: первоначально формируется предметная модель среды (нанесение контура протяженных источников, следов движения пробных зарядов, их классификация и пр.), а затем раскрываются закономерности ее построения. Особенно важно то, что исходный предмет квазиисследования (общее поле протяженных зарядов) преобразуется *самими* учащимися в другой (систему невзаимодействующих протяженных зарядов и т. д.).

Основным условием изучения понятия является моделирование не только его содержания в предметной среде, но и необходимых для раскрытия данного содержания *преобразований* этой среды. Последнее становится необходимым условием решения задачи в случае несоответствия результата преобразований новой игровой цели. Например, перемещение мишени на другую позицию и стремление учащихся на основе уже использованной модели своих действий реализовать новую игровую цель — поразить мишень — приводит к необходимости преобразовывать свои действия. В данных условиях формируется и новое представление о предметной среде. Важную роль в этом формировании играют контроль и оценка способов моделирования самим учащимся.

Как уже отмечалось, пока в процессе игры не возникает необходимости изменения свойств заданной среды, учащиеся не обращают внимания на эффекты, связанные с ее преобразованием. Чтобы они осознали значимость этих эффектов, требуется провести специальную поисковую работу по обнаружению тех или иных способов преобразования среды. Зафиксировав нужный эффект, учащиеся должны изменить схему своих действий, направив последние на усиление этого эффекта. При этом они создают новый способ изменения объекта.

Особое значение для разрабатываемых компьютерных моделей имеет преобразование изучаемых предметных сред как некоторых целостностей. Существующие формы анализа не позволяют выделять связи между элементами. Характерным примером такого подхода в рамках данного курса был бы переход от изучения общего поля протяженных зарядов сразу к изучению полей отдельных протяженных зарядов. При нашем подходе, учащиеся, обнаружив протяженные заряды на экране, фиксируют связи между ними; изменяя расстояние между источниками, преобразуют заданное целое (общее поле) в систему невзаимодействующих элементов (протяженных источников). При этом они фиксируют и свое преобразование (разведение зарядов), и изменение силовых линий.

При обучении с помощью компьютера организация действий обучаемого специфична, поскольку он имеет дело не с реальной предметной средой, а с ее компьютерной моделью (квазипредметностью). Все свойства модели должны быть заранее заложены в программу. В частности, в программе должны быть полностью определены все преобразования модели и способы осуществления вызова их учащимися. Введение новой операции в процессе обучения является весьма существенным моментом построения обучающих программ. Проблема однако заключается в том, чтобы учащийся не просто использовал заданную операцию, а *самостоятельно ввел* ее в схему, соответствующую содержанию решаемой задачи.

Нами предлагаются четыре возможных пути решения данной проблемы, каждый из которых имеет свои ограничения.

Первый путь может быть условно назван «скрытые операции». Суть его заключается в следующем. Все необходимые операции исходно заложены в компьютерную модель, но вначале учащемуся доступна только их часть. На основе этой части операций выполняется некоторая последовательность заданий. Когда же учащийся сталкивается с заданием, выполнить которое невозможно, используя только данное множество операций, и осознает это, то по указанию учителя он осваивает не известные ему ранее операции.

К достоинствам этого пути можно отнести управляемость и контроль процесса обучения,

а также простоту и ясность реализуемых методических схем.

Недостатки состоят в следующем:

обучающие программы неудобны в эксплуатации, поскольку требуют постоянного вмешательства учителя;

добавление новых операций является невольной подсказкой дальнейших действий (операция в данном случае выдается учащемуся готовой в тот момент, когда она понадобилась);

отсутствуют объективные критерии определения того, понимает ли учащийся необходимость выполнения новой операции;

требуется постоянное дополнительное инструктирование учащегося по пользованию программой.

Наш опыт, полученный в ходе разработки рассматриваемой компьютерной модели, показывает, что «скрытые операции» могут быть эффективно освоены учащимися. Использовать этот путь наиболее целесообразно при составлении заданий в игровой форме. Однако целостность курса распадается на отдельные фрагменты, а это является препятствием при формировании обобщенной картины учебного предмета.

Второй путь достаточно близок к первому и может быть условно назван «множественным выбором». Полная совокупность операций так же, как и в предыдущем случае, изначально закладывается в систему, но уже изначально является доступной обучаемому. Операции объясняются учащемуся в определенном порядке: сначала, наиболее подробно,— простые операции, затем, менее подробно,— сложные. При этом последовательность заданий строится таким образом, что при выполнении первых требуется использовать начальные (простые и понятные) операции. Правомерно ожидать, что учащийся будет пользоваться ими до тех пор, пока не возрастет сложность заданий и не появится необходимость осваивать другие операции.

Достоинствами этого пути является следующее:

предоставляется (хотя и формально) полная модель изучаемого предмета;

учащийся имеет значительно больше возможностей (по сравнению с предыдущим путем) в выборе и варьировании своих действий, что стимулирует его самостоятельность;

обучающая программа приобретает свойство внутренней замкнутости, т. е. способна функционировать без вмешательства учителя.

Среди недостатков этого пути отметим следующие:

значительно усложняется начальное вхождение учащегося в предметную область;

при выполнении заданий появляются отвлекающие факторы, связанные с попытками применения, даже в тех случаях, где в них нет необходимости, новых, неосвоенных операций (в связи с этим ослабляется контроль за процессом обучения);

возникают дополнительные ограничения на методические схемы при выборе заданий, поскольку нередко применение сложного действия (случайно и без надобности открытого учащимися) может полностью обессмыслить некоторые простые начальные задания, включенные в изучаемый курс.

Третий путь введения системы операций можно назвать «конструированием». Суть этого пути заключается в предоставлении учащемуся системы элементарных операций, из которых могут быть «сконструированы» действия (в простейшем случае — объединение операций в последовательность), необходимые для моделирования и анализа содержания данной предметной области. Условием успешного выполнения каждого конкретного задания является умение сформировать из элементарных операций новые действия (или выбрать необходимые из прежних).

Такой путь позволяет моделировать более сложные явления. Он обеспечивает учащемуся творческую свободу действий в рамках изучаемого предмета.

Среди недостатков этого пути следует отметить следующие:

наличие дополнительного механизма для синтеза действия из операций, весьма сложного как для реализации (фактически это задача, приближающаяся к разработке особого языка), так и для освоения учащимися. В ряде случаев подобный механизм превосходит по сложности саму предметную область;

далеко не всякая предметная область позволяет выделить подходящий набор операций и несложные способы их комбинирования;

трудно управлять распределением смысловой нагрузки в операциональном и предметном плане, а значит, гарантировать правильное усвоение материала.

Четвертый путь, которого мы старались придерживаться при разработке компьютерных моделей,— это путь так называемого «подавленного эффекта». Суть его заключается в следующем. Учащемуся изначально предоставляется как предметная область, так и полная система действий. Всякое действие в такой системе имеет множественный эффект, т. е. представляет собой некоторый вектор (спектр воздействия). Однако начальные (простейшие) задания подобраны так, что многие аспекты каждого конкретного действия оказываются почти незаметными,

«подавленными». И лишь по мере освоения учащимся предметной области эти скрытые эффекты начинают проявляться в заметной степени.

Достоинства данного пути следующие:

способ наиболее эффективен применительно к тем предметным областям, которые представлены максимально правдоподобной моделью;

поиск и выделение новых аспектов действия побуждает учащегося более интенсивно соотносить действия с содержанием предмета;

контроль за продвижением учащегося в предметной области обеспечивается постепенным разворачиванием действия и предмета по мере освоения материала.

Недостатки указанного пути заключаются прежде всего в сложности как построения, так и реализации самого компьютерного курса. Кроме того, может быть ослаблена по сравнению с третьим путем фиксация учащимся способов взаимодействия с предметной средой.

Вероятно, существуют еще и другие пути, а также возможны комбинации изложенных выше. Каждый из них нуждается в специальной психологической и методической оценке, в определении критериев выбора того или иного пути применительно к конкретному учебному предмету, а также возрасту учащихся.

Вместе с тем широкие возможности электронно-вычислительных устройств, включая компьютер как средство моделирования естественных и искусственных объектов, позволяют более полно реализовать принципы и методы развивающего обучения. Использование компьютерных систем учебного моделирования открывает новые перспективы в организации учебной деятельности по типу исследовательской и исследовательско-конструкторской. Положительные следствия такой организации весьма существенны. Это повышение мотивационной насыщенности учебного процесса, стимуляция выработки исследовательских и конструкторских навыков, снижение вероятности формального усвоения знаний.

Рекомендуемая литература

1. *Агеев В. В., Мульдаров В. К., Нежнов П. Г., Рубцов В. В.* К вопросу об использовании компьютера в учебном процессе (психологический аспект) // Психологические проблемы создания и исследования ЭВМ. М., 1985.

2. *Давыдов В. В.* Проблемы развивающего обучения. М., 1986.

3. *Давыдов В. В., Маркова А. К.* Концепция учебной деятельности школьников // Вопросы психологии. 1981. № 6.

4. *Ричмонд У. К.* Учителя и машины. М., 1968.

5. *Рубцов В. В.* Организация и развитие совместных действий у детей в процессе обучения. М., 1987.

6. *Столяров Л. Н.* Обучение с помощью машин. М., 1965.

7. *Талызина Н. Ф.* Теоретические проблемы программированного обучения. М., 1969.

8. *Papert S.* Mindstorms: Children, Computers and Powerful Ideas. N. Y., 1980.

Л. ПАВЛОВИЧ, О. ПАВЛОВИЧ

Применение ЭВМ при контроле усвоения знаний

Одним из существенных элементов процесса обучения является проверка знаний, умений и навыков, приобретаемых учащимися. Разработка оперативной системы контроля, позволяющей объективно оценивать знания учащихся, выявляя имеющиеся пробелы и определяя способы их ликвидации,— одно из условий совершенствования процесса обучения.

Как показала практика, несмотря на большое число дидактических материалов и методических рекомендаций по различным учебным предметам, адресованных преподавателям, многие из них испытывают большие трудности в организации контроля усвоения знаний учащихся. Так, опрос преподавателей химии СПТУ Татарии показал, что 96,8 % из них испытывают затруднения в организации постоянного контроля знаний слабоуспевающих учащихся; 80,3 % — в организации проверки профессионально значимых результатов обучения: знаний и умений политехнического и профессионально-прикладного характера; 30,3 % не могут достаточно четко дифференцировать и индивидуализировать контроль знаний; 13,3 % испытывают затруднения в планировании своей деятельности при организации контроля.

Затруднения, которые испытывают преподаватели при традиционных способах контроля усвоения знаний, на наш взгляд, частично или полностью можно устранить при использовании средств программированного контроля — ЭВМ и контролирующих

программ. Наиболее пригодны для этой цели персональные и микро-ЭВМ.

В данной статье мы представляем универсальную программу*, предназначенную для контроля знаний учащихся по различным дисциплинам. Программа может использоваться на различных этапах урока и на разных уровнях обучения: при текущем, тематическом и итоговом контроле знаний. Она работает в диалоговом режиме и не требует от преподавателей и учащихся специальной подготовки. Кроме средств контроля знаний в ней заложен ряд сервисных возможностей по формированию, коррекции и записи билетов и программ на магнитную ленту.

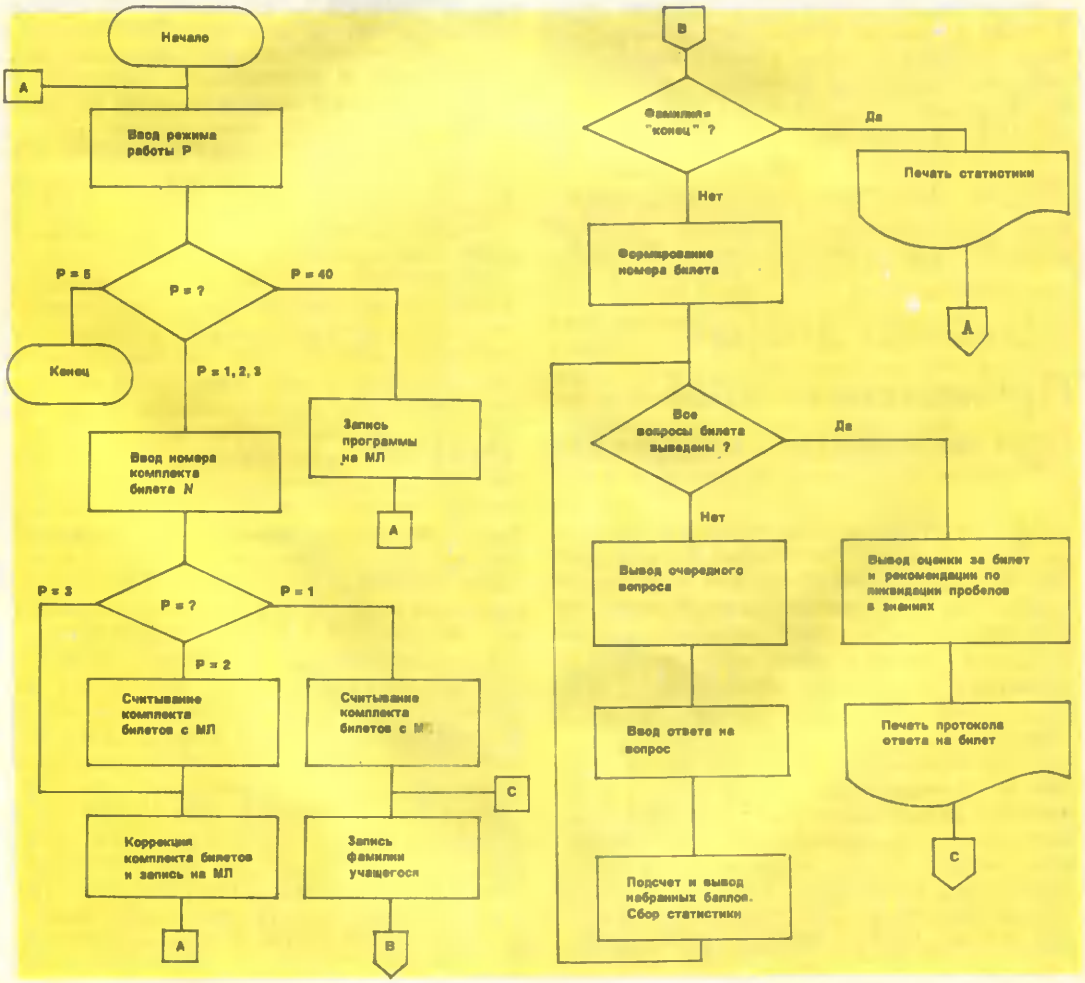
* Программа разработана на языке BASIK BLUS и реализована на ЭВМ «Электроника Д3-28». Она прошла апробацию на материале химии в СПТУ и при изучении элементной базы вычислительной техники в Казанском авиационном институте.

Для работы программы используются микро-ЭВМ, алфавитно-цифровой дисплей, печатающее устройство, интерпретатор с языка BASIK BLUS.

Программа состоит из двух частей: постоянной части, в которой производится вся обработка, и переменной, хранящей контролирующие тесты. Содержание переменной части программы зависит от учебного предмета. Все контролирующие вопросы разбиваются на четыре блока по уровню сложности.

I уровень — вопросы на распознавание. Их удобно представлять альтернативными тестами, предусматривающими ответы типа «да — нет», «правильно — неправильно», или тестами с выборочными ответами. II уровень — вопросы на воспроизведение или решение задачи. Они представляются тестами со свободным (конструктивным) ответом или с множественным выбором его.

III уровень — вопросы на применение



знаний при частично измененной ситуации или решение нетиповой задачи, которая после небольшого преобразования сводится к типовому случаю. Их удобно представлять тестами со свободным ответом, а также тестами с выборочными ответами и с выборочными пояснениями к ним. IV уровень — вопросы на творческое применение знаний, решение задач, не сводящихся к одному определенному типу. Представляются так же, как и вопросы III уровня.

Ответы на вопросы условно оцениваются следующим образом: I уровень — 0 или 1 балл; II уровень — 0, 1 или 2 балла; III уровень — 0, 1, 2 или 3 балла; IV уровень — 0, 1, 2, 3 или 4 балла.

Конструктивный ответ представляет собой строку произвольных символов. Эталон правильного ответа может задаваться в виде основы слова или фразы, что позволяет исключить влияние грамматических ошибок или возможных неточностей ответа на оценку.

Вопросы каждого уровня разбиты на тематические группы. По каждой группе вопросов указываются соответствующие номера параграфов в учебнике, номера задач и упражнений, которые следует дополнительно выполнить учащимся в случае неудовлетворительного ответа.

Работа с программой организована следующим образом. После ввода программы с магнитной ленты в машину преподаватель в ответ на запрос программы сообщает дату (число, месяц, год), и программа выводит текст: «Укажите режим работы: контроль знаний — режим 1; коррекция билетов — режим 2; формирование комплекта билетов — режим 3; запись программы на магнитную ленту — режим 40 (число 40, а не 4, выбрано для снижения вероятности ошибочного набора этого режима); конец работы — режим 5».

При выборе режима программа запрашивает номер формируемого комплекта билетов, название курса и максимальное число баллов за ответ. При компоновке билетов необходимо следить, чтобы сумма баллов за правильные ответы на вопросы одного билета в комплекте была постоянной.

В зависимости от цели урока и этапа, на котором осуществляется контроль знаний, преподаватель может формировать билеты с различным количеством вопросов разного уровня сложности, выбирать более или менее жесткие критерии оценки знаний учащихся. Например, при контроле начального усвоения учебного материала, цель которого — выяснить степень понимания данного материала основной массой учащихся, на этапе формирования новых знаний преподаватель может составить билеты из вопросов I и II уровней сложности. Допустим, такие билеты включают пять вопросов: два вопроса I уровня и три — II. Максимальное число баллов за ответы — 8 (два вопроса I уровня по 1 баллу и три вопроса II уровня по 2 балла). Поскольку проверка проводится сразу после объяснения учебного материала и критерии оценки знаний менее жесткие, оценки могут быть следующие: 7 баллов за отличный ответ, 5 — за хороший, 3 — за удовлетворительный. Полученная информация позволяет преподавателю выявить вопросы, к которым необходимо вернуться, так как они вызвали затруднения у большинства учащихся.

При контроле знаний на этапе формирования умений и навыков в билеты включаются вопросы всех уровней сложности. Например, в билет включены четыре вопроса каждого уровня сложности. Максимальное число баллов за такой ответ — 10. Преподаватель определяет число баллов за отличный, хороший и удовлетворительный ответ.

Определив степень усвоения конкретного материала каждым учащимся, преподаватель получает возможность дифференцировать тематический контроль. Допустим, он создает три комплекта билетов для трех групп учащихся (сильной, средней и слабой). Комплекты могут отличаться уровнем сложности вопросов и критериями оценки знаний. Организованная таким образом проверка знаний помимо контролирующей выполняет и стимулирующую функцию, позволяя слабым учащимся поверить в свои силы. Систематический контроль, предусматривающий и рекомендации (которые выдает программа) по ликвидации пробелов в знаниях, способствует повышению уровня знаний отстающих.

Работая в режиме формирования комплекта билетов, преподаватель по запросам программы указывает номер билета и номера вопросов. Сформированный комплект билетов программа (номера билетов и номера вопросов в них) выводит в табличной форме.

При работе в режиме 2 (коррекция билетов) с магнитной ленты вводится ранее сформированный комплект билетов. Затем, по запросу программы, преподаватель вводит номер билета и меняет в нем последовательность или число вопросов. Коррекция билетов необходима в тех случаях, если какие-то вопросы оказались непосильными или слишком легкими. Этот режим работы программы пригоден также для подготовки к дифференцированной проверке знаний. Дифференциация, как было указано выше,

осуществляется и при изменении критериев оценки знаний, которое также можно произвести в режиме коррекции билетов.

При выборе режима 1 программа запрашивает фамилию контролируемого. Затем машина произвольно выдает вопросы билета. После ответа на вопрос сообщается количество набранных баллов, а по окончании ответа на билет — суммарное число баллов, оценка и комментарии по ответу. Результаты сразу же фиксируются в протоколе. Протокол содержит дату (число, месяц, год), фамилию опрашиваемого, номер билета, число баллов за билет, число баллов за первый вопрос, за второй вопрос и т. д.

Комментарии, выдаваемые на экран дисплея, могут иметь следующий текст: «Молодец! Ваши знания по теме «отличные», «Вы хорошо усвоили материал по теме». В случае неудовлетворительной оценки учащиеся получают указания по повторению материала. Например: «Вы недостаточно усвоили материал о... Советую повторить по учебнику... §... и выполнить упражнение №... на стр...»

Для повышения объективности контроля в случае получения удовлетворительной или хорошей оценки учащемуся предоставляется право повторного ответа. При этом на экране появляется текст: «Если считаете, что Вам не повезло, можете ответить еще на один билет, но учтите, что в зачет пойдет последняя оценка. Рискнете?»

Опросив всех учащихся, преподаватель на запрос фамилии вводит слово «конец». По этому сигналу распечатываются статистические данные результатов контроля: коэффициент усвоения знаний (по вопросам каждого уровня) и общий коэффициент усвоения знаний группой учащихся по формуле $K_у = \frac{1_о}{1_а} \cdot 100\ \%$, где $1_о$ — количество полученных баллов; $1_а$ — максимальное число баллов.

В протоколах фиксируется уровень усвоения знаний (по мере прохождения учебного материала) как отдельных учащихся, так и группы в целом. Это позволяет корректировать билеты с учетом достигнутого уровня.

Оперативность информационного обеспечения при машинном варианте контроля усвоения знаний дает возможность планировать деятельность преподавателя, дифференцировать проверку, осуществлять систематический контроль, совмещать контроль усвоения знаний слабоуспевающих с ликвидацией пробелов в их знаниях.

Универсальность программы позволяет использовать ее для контроля знаний по различным учебным предметам, специфика которых проявляется в содержании текстовых заданий.

Роль компьютеров на АЭС

Знаете ли вы, что ЭВМ на атомных электростанциях выполняют сейчас подсобную, в общем-то, функцию — собирают информацию, обрабатывают ее и предоставляют человеку — оператору, который управляет работой АЭС? Конечно, ядерный реактор автоматизирован, как и всё, что вокруг него, но автоматика эта аналоговая, не использующая то, что мы привыкли называть электронно-вычислительной техникой. Ядерная энергетика развивалась быстрее электроники, и прототипы эксплуатируемых сейчас серийных энергоблоков АЭС создавались еще до появления современных ЭВМ.

Сейчас создаются системы, в которых управляющие вычислительные комплексы будут не только снабжать оператора информацией и давать ему советы-подсказки, но и непосредственно воздействовать на управляющие органы агрегатов. Такая цель поставлена потому, что анализ ситуации показал: интегрированная многокомпонентная стратегия управления даст лучшие результаты, чем набор индивидуальных аналоговых стратегий. Преимущества «цифрового» управления скажутся в повышении надежности (например, за счет регистрации неисправностей, калибровки, оценки измерений датчиков) и эффективности (в частности, благодаря возможности воспроизводить заранее рассчитанные движения регулирующих стержней реактора, подчиняющиеся сложным алгоритмам) атомных электростанций.

Когда важна скорость...

Сигнал «SOS»! Спасатели спешат на помощь...

Но даже если координаты кораблекрушения известны, до их прибытия проходит время.

Направление и сила ветра, морские течения, приливы и отливы, местные условия — все влияет на форму и размер зоны поисков. Чтобы рассчитать ее вручную, нужно 2—3 часа, и точность при этом оставляет желать лучшего.

Недавно создано соответствующее программное обеспечение, с помощью которого ЭВМ не только точно вычислит границы зоны за несколько минут, но и даст рекомендации по тактике использования имеющихся спасательных средств — самолетов с аппаратурой обнаружения, вертолетов, катеров и т. п.

В Министерстве просвещения СССР

О преподавании курса «Основы информатики и вычислительной техники» в 1987/88 учебном году

Методическое письмо

В 1987/88 учебном году количество учебных часов, выделяемых на изучение курса «Основы информатики и вычислительной техники» в IX и X классах, сохраняется то же, что и в 1986/87 году (34 ч — IX и 34/68 ч — X). Рекомендации по преподаванию курса X класса в объеме 68 ч приведены в методическом письме «О преподавании курса «Основы информатики и вычислительной техники» с примерным планированием учебного материала X класса для школ, в которых изучение курса проводится в объеме 68 часов». Письмо опубликовано в журнале «Информатика и образование» № 3, 1987 г.

В тех школах, где имеется вычислительная техника с соответствующим программным обеспечением и есть учитель, получивший необходимую подготовку, возможна организация занятий учащихся на ЭВМ уже с IX класса. Для этого рекомендуется использовать программу конкурсного учебника, опубликованную в журналах «Информатика и образование» № 1 за 1986 г., «Математика в школе» № 3 за 1986 г. и «Микропроцессорные средства и системы» № 2 за 1986 г. и Систему программирования на основе алгоритмического языка курса ОИВТ (Е-практикум). Практические занятия организуются в рамках учебных часов, предусмотренных учебным планом. Переход школы на преподавание курса по конкурсной программе необходимо согласовать с районным отделом народного образования.

Ниже приводятся рекомендации по преподаванию курса «Основы информатики и вычислительной техники» в предстоящем учебном году.

IX класс

Анализ преподавания курса в 1985/86 и 1986/87 учебных годах показывает, что у многих учащихся не удается сформировать такие важнейшие понятия информатики, как алгоритм, исполнитель и система его команд, величины и операции над ними. В результате школьники плохо воспринимают теоретическую часть курса «Основы информатики и вычислительной техники», а их знания, умения и навыки оказались поверхностными и непрочными.

Для устранения отмеченных недостатков предлагается перераспределить количество учебных часов на изучение тем: «Алгоритм. Алгоритмический язык», «Алгоритмы работы с величинами». При этом количество разбираемых в классе примеров может быть увеличено за счет использования упражнений из раздела «Алгоритмы работы с графической информацией». Материал этого раздела рекомендуется (как и в прошлом году) рассмотреть в начале учебного года.

Рекомендуется увеличить число упражнений на исполнение алгоритмов. Следует акцентировать внимание школьников на изменении значений величин при выполнении различных команд. Использование таблиц значений по форме, приведенной в учебном пособии, не является обязательным. Учитель может по своему усмотрению использовать и другие способы записи результатов исполнения алгоритмов. При разборе несложных алгоритмов возможно их исполнение и без построения таблиц значений.

Учитывая рекомендации по планированию учебного материала IX класса, изложенные в методическом письме «О преподавании курса «Основы информатики и вычислительной техники» в 1986/87 учебном году», а также перераспределение материала, о котором говорилось выше, предлагается следующее планирование.

Примерное планирование учебного материала для IX класса
(1 ч в неделю, всего 34 ч)

Введение (2 ч)

Роль ЭВМ в современном обществе — *1 ч*
Первоначальные сведения об ЭВМ — *1 ч*

Алгоритмы. Алгоритмический язык (6 ч)

Понятие алгоритма. Формальное исполнение алгоритма. Общие правила алгоритмического языка — *1 ч*

Алгоритмы работы с графической информацией — 1 ч

Составные команды. Решение задач. Самостоятельная работа — 4 ч

Алгоритмы работы с величинами (8 ч)

Величины. Заголовок алгоритма — 2 ч

Промежуточные величины. Присваивание значений. Отношение между величинами в качестве условий — 2 ч

Исполнение алгоритма — 2 ч

Решение задач — 1 ч

Контрольная работа № 1 — 1 ч

Табличные величины. Алгоритмы работы с табличными величинами (6 ч)

Табличные величины. Решение задач — 2 ч

Поиски заданного элемента в линейной таблице — 1 ч

Поиск наименьшего элемента в линейной таблице — 1 ч

Решение задач. Самостоятельная работа — 2 ч

Вспомогательные алгоритмы (7 ч)

Понятие вспомогательного алгоритма. Последовательное построение алгоритма. Упорядочение линейной таблицы — 3 ч

Решение задач — 3 ч

Контрольная работа № 2 — 1 ч

Построение алгоритмов для решения задач (5 ч)

Этапы решения задач с использованием ЭВМ — 1 ч

Построение алгоритмов для решения задач из курсов математики и физики. Решение задач — 4 ч

X класс

Теоретическая часть курса X класса развивает основные понятийные линии, начатые в предыдущем учебном году. Продолжается изучение алгоритмов, их свойств и методов построения. Расширяются сведения учащихся о величинах и операциях над ними. При этом школьники привыкают к специальным обозначениям и правилам записи, используемым в языках программирования, подготавливаются к освоению применяемых на практике языков программирования.

В предстоящем учебном году представляется целесообразным переставить некоторые темы учебного материала.

Рекомендуется начать курс X класса с изучения учащимися раздела II «Знакомство с программированием», а раздел I «Устройство ЭВМ» перенести в конец курса. Это позволит естественно продолжить изучение основных понятий информатики, используя аппарат алгоритмического языка.

Материал § 8 «Вспомогательные алгоритмы вычисления значений функций» может быть упрощен за счет исключения из рассмотрения рекурсивных алгоритмов (с. 49—51 учебного пособия, ч. II).

Выбор языка программирования и соответствующее распределение учебных часов по данной теме с учетом имеющихся возможностей организации практических занятий предоставляются учителю. В случае невозможности организовать занятия школьников на ЭВМ наиболее целесообразным представляется изложение языка Рапира как более близкого к учебному алгоритмическому языку. В этом случае следует придерживаться распределения учебного материала, приведенного в методическом пособии, увеличив количество часов на решение задач.

Рекомендуется уделить большее внимание теме «Роль ЭВМ в современном обществе. Перспективы развития вычислительной техники», увеличив количество часов на ее изучение до 4. Знакомство учащихся с областями применения средств вычислительной техники не должно ограничиваться уроками, посвященными этому вопросу. Целесообразно поручать учащимся в качестве домашних заданий подготовку рефератов, обзоров печати (публикаций журнала «Информатика и образование») и т. п. по отдельным направлениям использования вычислительной техники и практиковать краткие доклады учащихся на большинстве уроков.

Изучение материала курса заканчивается темой «Устройство ЭВМ». Рекомендуется упростить ее изложение, исключив рассмотрение материала о командах условного и безусловного перехода. При изучении темы учащиеся должны ознакомиться с командами процессора. При этом заучивать эти команды школьникам не обязательно.

Примерное планирование учебного материала для X класса

(1 ч в неделю, всего 34 ч)

Знакомство с программированием (18 ч)

Алгоритмический язык (10 ч)

Команда выбора. Решение задач — 2 ч

Команда повторения с параметром. Решение задач — 3 ч

Вспомогательные алгоритмы вычисления значения функции. Решение задач — 2 ч

Алгоритмы работы с литерными величинами. Решение задач — 2 ч

Контрольная работа № 1 — 1 ч

Язык программирования (включая решение задач и контрольную работу № 2) — 8 ч

Роль ЭВМ в современном обществе (4 ч)

Краткая история вычислительной техники — 2 ч

Программное обеспечение ЭВМ — 2 ч

Устройство ЭВМ (8 ч)

Общая схема устройства ЭВМ. Основной алгоритм работы процессора — 4 ч

Представление информации в ЭВМ. Физические принципы работы ЭВМ. Решение задач. Самостоятельная работа — 4 ч

Экскурсия на вычислительный центр — 4 ч

Организация практических занятий учащихся на ЭВМ

Для школ, имеющих возможность организовать систематические занятия школьников на ЭВМ, программой курса «Основы информатики и вычислительной техники» предусмотрено выделение дополнительных 34 ч для практической работы учащихся на ЭВМ.

Планирование практических занятий и методи-

ческие рекомендации по их проведению приводятся в методическом письме «О преподавании курса «Основы информатики и вычислительной техники» с примерным планированием учебного материала X класса для школ, в которых изучение курса проводится в объеме 68 часов».

При организации практических занятий рекомендуется обратить внимание на следующее.

Согласно временным санитарно-гигиеническим нормам, действующим с 1.09.87, длительность работы школьников за видеотерминалами школьных ЭВМ («Агат», КУВТ «Ямаха», КУВТ-86 и др.) в режиме набора информации на клавиатуре ограничивается 20 мин в неделю.

Готовясь к проведению практических занятий, учитель должен планировать использование ЭВМ не менее половины урока в режиме демонстраций. Остальное время занятий учащихся в кабинете вычислительной техники должно быть посвящено работе с готовым программным обеспечением.

В условиях недостаточного оснащения школ компьютерами рекомендуется организовывать на базе школьных ЭВМ межшкольные кабинеты вычислительной техники, в задачи которых будут входить:

организация практических занятий учащихся близлежащих общеобразовательных школ;

проведение кружковой работы и факультативных занятий с учащимися других возрастных групп;

подготовка учителей других предметов к использованию ЭВМ в своей работе в рамках 36-часового курса (программа курса опубликована в журнале «Информатика и образование» № 2 за 1986 г.).

Для этих целей можно использовать программное обеспечение, поставляемое централизованно для различных видов техники.

Программы факультативных курсов могут разрабатываться учителями-практиками с последующим их обсуждением и утверждением педагогическими советами школ.

Как показывает практика, наиболее эффективно курс изучается в тех школах, где практические занятия на ЭВМ ведут школьные учителя, преподающие теоретическую часть курса.

При утверждении других вариантов программы преподавания курса в условиях кабинета вычислительной техники районным органам народного образования следует обращать внимание на то, чтобы в ее содержание включались основные темы действующей программы курса*.

При подготовке к проведению занятий по курсу «Основы информатики и вычислительной техники» учитель может использовать материалы методического пособия для преподавателей «Изучение основ информатики и вычислительной техники», части I, II (М.: Просвещение, 1985, 1986), а также методические рекомендации и наборы упражнений для учащихся, приведенные в ряде публикаций в «Учительской газете», журналах «Информатика и образование» (рубрики «Методика обучения», «КВТ»), «Математика в школе», «Вечерняя средняя школа».

При проверке и оценке знаний и умений школьников можно руководствоваться рекомендациями соответствующего методического письма, опубликованного в журнале «Информатика и образование» № 1, 1987 г.

Центральное телевидение в наступающем учебном году продолжит цикл передач по курсу IX и X классов в помощь учителю. Цикл будет включать в себя повторение лучших передач из показанных в 1986/87 учебном году и новые передачи, посвященные организации практических занятий (тематика передач будет опубликована в «Учительской газете» в августе 1987 г.).

В 1986 г. в помощь учителю информатики выпущена серия таблиц «Алгоритм и алгоритмический язык. Алгоритмы работы с величинами».

В 1987 г. в школы поступят:

печатные пособия «Структура и принцип работы ЭВМ», «Перспективы развития ВТ в свете решений XXVII съезда КПСС»;

диапозитивы «Устройства ввода и вывода информации»;

кинофильмы «Роль ЭВМ в современном обществе», «Первоначальные сведения об ЭВМ»;

транспаранты «Устройство и принцип действия ЭВМ», «Представление линейного алгоритма», «Представление команды ветвления», «Представление команды повторения».

Приложение

Примерные варианты контрольных работ по курсу «Основы информатики и вычислительной техники»

IX КЛАСС

Контрольная работа № 1

Задание 1. Дописать заголовок алгоритма, описать типы величин.

Вариант 1

```
алг ПРОВЕРКА (  )
    арг
    рез
нач      d
  d:=a+b
    если  a>b
      то z:=«сложение»
         y:=10+d
      иначе z:=«вычитание»
            y:=10—d
    все
    y:=y·d
кон
```

Вариант 2

```
алг ПРОВЕРКА (  )
    арг
    рез
нач      d
  d:=10·b
    если x=«формула 1»
      то a:=7·d
         v:=«умножение»
```

* Программа для средних учебных заведений «Основы информатики и вычислительной техники». М.: Просвещение, 1985.

иначе $a:=7+d$
$v:=$«сложение»
все
$a:=(a\cdot b)-d$
кон

Задание 2. Исполнить алгоритм из задания 1 для

Вариант 1

а) $a=5$, $b=3$;
б) $a=0$, $b=6$.

Вариант 2

а) $x=$«формула 1», $b=5$;
б) $x=$«формула 2» $b=1$.

Задание 3. Составить алгоритм, при исполнении которого вычерчивается фигура

Вариант 1 *Вариант 2*

Контрольная работа № 2

Задание 1. Запишите данный алгоритм с использованием команды повторения.

Вариант 1

```
алг ВЫЧИСЛЕНИЕ (вещ S, вещ таб A[1:5])
    арг A
    рез S
нач
    S:=0
    S:=S+A[1]-1
    S:=S+A[2]-2
    S:=S+A[3]-3
    S:=S+A[4]-4
    S:=S+A[5]-5
кон
```

Вариант 2

```
алг ВЫЧИСЛЕНИЕ (вещ X, вещ таб A[1:5])
    арг X
    рез A
нач
    A[1]:=X+1
    A[2]:=X+2
    A[3]:=X+3
    A[4]:=X+4
    A[5]:=X+5
кон
```

Задание 2. Известны значения элементов таблиц целых чисел A[1:5] и B[1:5]. Постройте алгоритм определения значений элементов таблицы X[1:5], если известно, что:

Вариант 1

$X[I]=3\cdot A[I]+0.5\cdot B[I]$

Вариант 2

$X[I]=2\cdot A[I]+B[I]-1$

Задание 3.

Вариант 1

Постройте алгоритм нахождения наибольшего из четырех чисел A, B, C, D. Использовать в качестве вспомогательного алгоритм БИД из библиотеки.

Вариант 2

Постройте алгоритм нахождения наибольшего общего делителя четырех натуральных чисел A, B, C, D. Использовать в качестве вспомогательного алгоритм НОД из библиотеки.

X КЛАСС

Контрольная работа № 1

Задание 1.

Вариант 1

Какие значения получат величины X1, X2, X3 после исполнения алгоритма, если A=«величина».

```
алг ПРЕОБРАЗОВАНИЕ (лит A, X1, X2, X3)
    арг A
    рез X1, X2, X3
нач
    X1:=A[1:2]
    X2:=A[7:8]
    X3:=X1+X2
кон
```

Вариант 2

Какие значения получат величины A, B, C после исполнения алгоритма, если X=«гиена».

```
алг ПРЕОБРАЗОВАНИЕ (лит A, B, X, цел C)
    арг X
    рез A, B, C
нач
    A:=X[1:2]
    B:=A+X
    C:=длин(B)
кон
```

Задание 2. Постройте алгоритм вычисления значений функции.

Вариант 1

$$y=\begin{cases}0, \text{ если } x<0\\ 2\cdot x, \text{ если } 0\leqslant x\leqslant 1\\ x^2+1, \text{ если } x>1\end{cases}$$

Вариант 2

$$y=\begin{cases}x^2, \text{ если } x<1\\ 1, \text{ если } x=1\\ 2x+3, \text{ если } x>1\end{cases}$$

Задание 3. Известны значения элементов таблиц целых чисел A[1:5] и B[1:5]. Постройте алгоритм определения значений элементов таблицы X[1:5], если известно, что:

Вариант 1

X[I]=A[I]+B[6—I]

Вариант 2

X[I]=A[6—I]—B[I]

Контрольная работа № 2

В качестве заданий рекомендуется использовать тексты работ, приведенные в методическом пособии «Изучение основ информатики и вычислительной техники», ч. II.

Возможные задания к самостоятельной работе по теме «Устройство ЭВМ»
(выбор заданий предоставляется учителю)

1. Постройте и исполните программу по заданному алгоритму:

Вариант 1

```
алг ЗАДАЧА (цел R0, R1, R2, R3)
   арг
   рез
нач
   R1:=0
   R1:=R1+R0
   R1:=R1+2
   R1:=R1—R2
   R3:=R1
кон
```

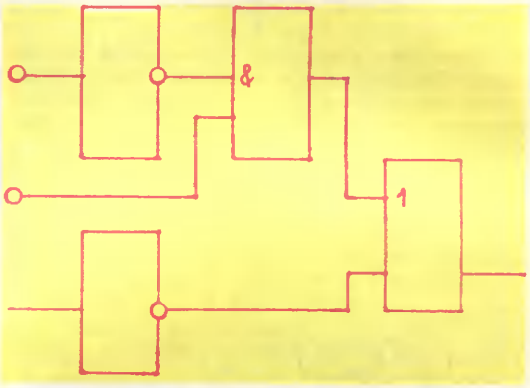
Вариант 2

```
алг ЗАДАЧА (цел R1, R2, R3, R4)
   арг
   рез
нач
   R2:=0
   R2:=R3—R4
   R2:=R2—1
   R1:=R2
кон
```

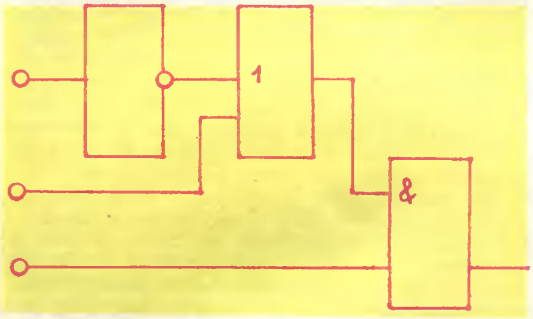
2. Запишите следующие числа
а) в двоичной системе:
вариант 1: 75, 137; *вариант 2:* 59, 105;
б) в десятичной системе:
вариант 1: 0101101, 1100111000; *вариант 2:* 11001101, 01110001100.

3. Составьте таблицу работы логической схемы:

Вариант 1



Вариант 2



Программировать по-новому

Сегодня Министерство промышленности средств связи является одним из крупнейших потребителей электронной техники. На ней базируются практически все вновь создаваемые системы и средства — от интегрированных сетей обмена данными до «электронных» АТС и кнопочных телефонов.

Но без программ ЭВМ мертва, и отрасль оказалась перед сложной задачей — в двенадцатой пятилетке, например, необходимо разработать в 3,5 раза больше программных средств, чем в одиннадцатой. Переманивать программистов? А что делать, когда необходимость в них увеличится еще больше?

Задачу помогла решить новая технология программирования, развиваемая в Институте кибернетики им. В. М. Глушкова АН УССР,— так называемая Р-технология. Она использует графическую форму представления алгоритмов и повышает производительность труда программиста в 2—7 раз.

Об организации подготовки педагогических кадров по основам информатики и вычислительной техники в институтах усовершенствования учителей

В соответствии с приказом Министерства просвещения СССР «О мерах по обеспечению компьютерной грамотности учащихся средних учебных заведений и широкого внедрения электронно-вычислительной техники в учебный процесс» в учебно-тематические планы курсов повышения квалификации учителей и других категорий работников просвещения введен раздел «Использование ЭВТ в учебно-воспитательном процессе». На его изучение отводится в зависимости от категории обучаемых от 6 до 12 часов. Однако, как показывает практика, данного объема учебного времени недостаточно для приобретения слушателями необходимых знаний и навыков общения с электронно-вычислительной техникой.

В связи с этим Министерство просвещения СССР инструктивным письмом «О подготовке руководящих и педагогических кадров народного образования по основам информатики и вычислительной техники» рекомендовало организовать дополнительную подготовку по специальной 36-часовой программе руководящих и педагогических кадров народного образования, кроме специализирующихся в области информатики, независимо от того, знакомился ли слушатель ранее с данной темой на курсах повышения квалификации.

Для дальнейшего повышения квалификации учителей информатики и вычислительной техники рекомендуется осуществить их дополнительную подготовку, исходя из местных условий, к 1987/88 учебному году по методике преподавания наиболее трудных вопросов курса «Основы информатики и вычислительной техники» IX—X классов.

Для определения содержания занятий можно воспользоваться тематическим планом и программой.

Обучение слушателей основам информатики и вычислительной техники предполагает активное использование новой формы занятий — практические занятия на ЭВМ, оплата которых производится в том же порядке, что и оплата занятий по анализу конкретных ситуаций, решению производственных задач и т. д. (п. 26 Инструкции о порядке исчисления заработной платы работников просвещения, утвержденной приказом Министерства просвещения СССР от 16 мая 1985 г. № 94).

Практические занятия слушателей на ЭВМ проводятся по подгруппам, численность которых определяется в соответствии с количеством рабочих мест в кабинете вычислительной техники при условии, что за одним дисплеем одновременно работает не более двух человек.

Тематический план и программа

Научно-методические основы курса «Информатика и вычислительная техника»*

Настоящая программа, рассчитанная на 72 часа, предназначена как для первоначальной подготовки учителей к преподаванию курса «Основы информатики и вычислительной техники» (I вариант), так и для продолжения непрерывной подготовки учителей, прошедших начальную подготовку по курсу ОИВТ в объеме 144 часов (II вариант). Эти варианты отличаются как по количеству часов, отводимых на изучение отдельных разделов программы, так и по направленности подготовки.

Проведение первоначальной подготовки (I вариант) должно обеспечить необходимый научный уровень учителей в области информатики и ВТ, подготовить их к проведению практических занятий по курсу ОИВТ в кабинетах вычислительной техники, обеспечить необходимый методический уровень преподавания курса. Особое внимание необходимо уделить усвоению учителями научного содержания курса, базирующегося на трех основных понятиях: информация, алгоритм, ЭВМ.

В процессе доподготовки (II вариант) больше внимания целесообразно уделить вопросам методики изложения тем и отдельных вопросов программы, а также особенностям комплектов учебной вычислительной техники (КУВТ) различных типов, имеющемуся программному обеспечению и методике его использования.

Тематический план

Формы проведения занятий:

л — лекции, где излагаются научно-методические основы данной темы;

с — семинарские занятия, на которых рассматриваются конкретные вопросы содержания и методики преподавания темы;

п — практические занятия, на которых формируется умение работать на ЭВТ.

---

* Разработано Московским городским институтом усовершенствования учителей. Рекомендовано Управлением информатики и вычислительной техники Министерства просвещения СССР.

№ п/п	Тема	Количество часов							
		I вариант				II вариант			
		Всего	л	с	п	Всего	л	с	п
1	Цели, задачи и первые итоги изучения курса ОИВТ	4	4	—	—	4	4	—	—
2	Кабинет вычислительной техники и программное обеспечение	10	2	—	8	20	10	—	10
3	Алгоритмический язык	24	8	8	8	16	4	6	6
4	Язык программирования	18	4	4	10	20	4	6	10
5	Устройство и принципы работы ЭВМ	12	8	—	4	8	4	—	4
6	Роль ЭВМ в обществе	4	4	—	—	4	4	—	—
	Всего	72	30	12	30	72	30	12	30

Программа

Цели, задачи и итоги изучения курса ОИВТ

Цели и задачи курса ОИВТ. Итоги изучения курса в IX—X классах. Различные варианты преподавания курса (без систематического использования вычислительной техники на уроках, с практическими занятиями на ПЭВМ, с использованием ПКМ). Анализ состояния преподавания курса, уровня знаний, умений и навыков учащихся. Нормативные документы, касающиеся вопросов компьютеризации школ. Задачи на новый учебный год.

Кабинет вычислительной техники и программное обеспечение

Комплекты учебной вычислительной техники (КУВТ-86, «Агат», «УК-НЦ», «Корвет», «Ямаха»). Обзор технических характеристик и методических возможностей различных классов: назначение локальной сети, языки программирования, программное обеспечение. Основные требования к оборудованию и эксплуатации кабинетов вычислительной техники. Организация технического обслуживания. Техника безопасности, санитарно-гигиенические и эргономические требования.

Назначение и состав программного обеспечения ЭВМ. Системное, базовое и прикладное программное обеспечение ЭВМ. Операционная система, ее назначение и состав. Операционная система школьных ЭВМ. Пересылка программ по локальной сети.

Базовое программное обеспечение ЭВМ: текстовый и графический редакторы, база данных, электронные таблицы.

Прикладное программное обеспечение ЭВМ. Использование прикладных программ для решения учебных и производственных задач.

Программное обеспечение, поддерживающее курс ОИВТ: клавиатурный тренажер, Е-практикум, ЭВМ-практикум.

Организация работы школьных и межшкольных кабинетов вычислительной техники. Методика проведения занятий в кабинетах вычислительной техники.

Кабинет вычислительной техники — центр урочной и внеурочной работы по компьютеризации школы. Организация общественно полезного, производительного труда учащихся по составлению прикладного программного обеспечения для КУВТ.

Алгоритмический язык

Алгоритмический язык как средство выражения и записи алгоритмов. Три типа алгоритмов: линейный, разветвляющийся, циклический. Основные средства алгоритмического языка: команды ветвления, выбора, повторения типа «пока» и «для», литерные и графические величины, табличные величины, вспомогательные алгоритмы. Последовательное построение алгоритма.

Методика изучения темы «Алгоритмический язык». Практикум по составлению алгоритмов. Реализация алгоритмов на ЭВМ с использованием Е-практикума.

Язык программирования

Сопоставление языков программирования (Бейсик, Фокал, Рапира, Паскаль). Различные структуры языков программирования, их достоинства и недостатки. Основные средства языка (типы переменных, операторы, функции, процедуры) на примере одного из вышеперечисленных языков.

Методика изучения языка программирования, реализация методического принципа «от алгоритма к программе».

Практикум по составлению программ. Составление программ для решения задач по обработке числовой, текстовой и графической информации на основе межпредметных связей с курсами математики, физики и других общеобразовательных дисциплин. Программы решения задач, обработки результатов лабораторных работ, контролирующие, обучающие, моделирующие программы по общеобразовательным дисциплинам.

Устройство и принципы работы ЭВМ

Структура ЭВМ. Физические принципы работы ЭВМ. Представление информации в ЭВМ. Логические операции и базовые логические функции «И», «ИЛИ», «НЕ». Конструирование элементов ЭВМ (триггер, сумматор) на основе базовых логических функций. Алгоритм работы процессора.

Методика изучения темы «Устройство ЭВМ». Реализация алгоритмов работы процессора на ЭВМ с использованием ЭВМ-практикума.

Роль ЭВМ в обществе

Развитие вычислительной техники: прошлое, настоящее, будущее. Роль ЭВМ в социально-экономическом развитии общества. Основные сферы применения ЭВМ: справочно-информационные системы и информационные сети, обработка текстов, системы машинной графики, программы для расчетов, станки с ЧПУ, роботы.

А. ШЕНЬ

Информатика в IX классе

Алгоритмы работы с величинами

Аргументы и результаты. До сих пор рассматривались алгоритмы управления исполнителем, действующим в какой-то обстановке и меняющим ее в ходе исполнения алгоритма (наливающим воду, переключающим электроприбор, рисующим на плоскости и т. д.). Теперь перейдем к алгоритмам, которые не производят каких-либо изменений в окружающем их мире, а лишь перерабатывают поступающую информацию и выдают результаты этой переработки.

Исполнителя такого алгоритма можно сравнить с экспертом, сидящим у себя в кабинете. Ему звонят по телефону и сообщают исходные данные, необходимые для работы (их называют аргументами). После этого он выключает телефон и начинает работать. Кончив работу, он вновь включает телефон и сообщает результаты своей работы.

Понятия аргументов и результатов алгоритма можно пояснить такими примерами.

1. Задача: найти наибольший общий делитель двух натуральных чисел. Аргументы: два натуральных числа. Результаты: натуральное число, являющееся их наибольшим общим делителем.

2. Задача: подсчитать количество букв «а» в слове. Аргументы: слово. Результаты: целое число, равное количеству букв «а» в этом слове.

3. Задача: вычислить квадратный корень. Аргументы: вещественное число. Результаты: вещественное число, являющееся квадратным корнем из исходного.

4. Задача: разделить одно натуральное число на другое с остатком. Аргументы: два натуральных числа (делимое и делитель). Результаты: два целых числа (частное и остаток).

5. Задача: сложить числа в таблице. Аргументы: таблица, заполненная вещественными числами. Результаты: вещественное число, равное сумме всех чисел таблицы.

6. Задача: перевести слово по русско-английскому словарю. Аргументы: русское слово. Результаты: английское слово, являющееся переводом русского.

7. Задача: найти центр окружности, проходящей через данные три точки. Аргументы: три точки плоскости. Результаты: точка плоскости, являющаяся центром искомой окружности.

Типы. В каждой из перечисленных задач аргументы и результаты — данные какого-то определенного типа. Нельзя искать, например, наибольший делитель двух слов, переводить вещественное число по русско-английскому словарю или строить центр окружности, проходящей через данные три числа. При записи алгоритмов мы будем указывать типы исходных данных (аргументов) и результатов, т. е. указывать, какие значения они могут принимать. В школьном курсе информатики рассматриваются четыре типа: целый, вещественный, натуральный и литерный. Данные целого, вещественного и натурального типов — соответственно, целые, вещественные и натуральные числа. Данные литерного типа — произвольные последовательности символов.

Понятие типа и его использование в УП вызывают много вопросов. Обсудим некоторые из них.

Продолжение. Начало см.: Информатика и образование. 1987. № 4.

1. Бывают ли другие типы, кроме перечисленных?

Прежде всего надо иметь в виду, что вопрос поставлен не вполне правильно. Типы, используемые в алгоритмическом языке, определены его разработчиками, так что спрашивать можно так: «Включены ли в алгоритмический язык учебника другие типы?» Да, включены. Это типы, связанные с табличными величинами (однако они плохо продуманы авторами, о чем мы еще будем говорить).

Другое возможное уточнение вопроса: «Возникает ли при решении задач необходимость в других типах, кроме приведенных в УП?» Да. Это видно уже из примера 7, где необходим тип «точка плоскости». Недаром во многих языках программирования имеются другие типы и даже средства, позволяющие программисту (составителю алгоритма) самому создать (определить) новый тип. Например, на Паскале можно написать

```
TYPE ТочкаПлоскости=
  RECORD
    Хкоорд: REAL;
    Укоорд: REAL
  END;
```

При этом определяется новый тип (TYPE), который называется «ТочкаПлоскости»; данное этого типа запись (RECORD) из двух вещественных (REAL) чисел: Хкоорд и Укоорд. После этого слово «ТочкаПлоскости» можно использовать как название типа наряду с типами REAL (вещ) и др.

2. В примере 2 (вычисление квадратного корня) аргументом может быть не любое вещественное число, а лишь неотрицательное. Можно ли использовать при составлении этого алгоритма тип вещ? Не нужен ли специальный тип неотрвещ (неотрицательное вещественное число) или что-то в этом роде?

Тип вещ использовать можно. Дело в том, что мы не предполагаем, что допустимы все исходные данные, относящиеся к указанному в алгоритме типу. Если бы это было не так, то алгоритм вычисления, например, значения выражения $1/(x^2-x-1)$ требовал бы специального типа «вещественное число, не равное $(1+\sqrt{5})/2$ и $(1-\sqrt{5})/2$» — ведь эти числа обращают знаменатель в нуль!

3. Если не предполагается, что все значения указанного в алгоритме типа допустимы как исходные данные, то зачем нужен тип нат? Ведь натуральные числа — часть целых, и любое данное типа нат можно считать данным типа цел. И зачем нужен тип цел, если целые числа — часть вещественных?

Ответ нужно начать с повторения уже сказанного: необходимость того или иного типа — вопрос удобства. Авторы УП сочли, что полезно ввести специальный тип «натуральные числа», поскольку ситуация, в которой какая-то величина может принимать только натуральные значения, часто встречается. Это решение, однако, оказалось спорным: в школьном курсе математики 0 не считается натуральным числом, поэтому тип нат оказывается малоприменимым. Уже в самом первом примере заголовка алгоритма (с. 31 УП) тип нат употреблен ошибочно: ведь и частное, и остаток при делении натуральных чисел могут быть равны нулю. Поэтому мы рекомендуем вообще отказаться от употребления типа нат.

Иначе обстоит дело с типами цел и вещ. Дело в том, что вещественные числа представляются в машине лишь приближенно (это неизбежно, так как вещественных чисел слишком много). Всякий, кто пробовал пользоваться калькулятором, знаком с неожиданностями типа $(1/3)\cdot 3=0.99999$, $10000+0.00001=10000$ и т. д.

Подобные «сюрпризы» могут нарушать работу составленных нами алгоритмов. Естественно желание гарантировать точность представления чисел и операций с ними, а в случае целых чисел это возможно. Поэтому, описывая аргумент алгоритма как цел, мы не только обязуемся давать исполнителю в качестве аргументов лишь целые числа, но и требуем от него точного выполнения операций над ними.

4. Зачем указывать тип заранее? Почему бы не разрешить одной и той же величине в одном случае принимать числовое значение, в другом — литерное и т. п.?

Такой подход действительно возможен и используется, например в языке программирования Рапира. Он имеет как преимущества, так и недостатки. Решение, принятое в УП (каждая величина имеет фиксированный тип), можно обосновать следующим образом. Алгоритмы не только записывают на алгоритмическом языке, но и читают, а при этом бывает полезно знать, значения какого типа может принимать данная величина. Если типы величин задаются заранее, то узнать это просто, если же типы величин могут меняться, то эта задача сильно усложняется. Кроме того, в случаях, когда важны быстрота выполнения программ и занимаемый ими объем памяти, предварительное указание типов дает возможность хранить их в памяти более экономно и выполнять операции с ними быстрее; при этом величины типа цел, как правило, занимают меньше места, чем типа вещ, и быстрее обрабатываются.

В завершение следует сказать, что все эти тонкости, скорее всего, не будут замечены школьниками. Это и хорошо — при первом знакомстве с алгоритмическим языком встают более насущные вопросы.

Исполнение алгоритмов

В УП предлагается изображать исполнение алгоритмов с помощью таблиц значений. Мы предложим другой способ, быть может, более удобный при работе в классе с доской и мелом. Разберем исполнение алгоритма примера 7.1 УП (решение квадратного уравнения). В нем есть такие величины:

аргументы a, b, c;
результаты x_1, x_2, y;
промежуточная величина D.

Для каждой из них нарисуем на доске рамку, внутри которой будем писать значение величины, а снаружи — имя. Как выглядит доска, подготовленная для разбора этого примера, изображено на рис. 1.

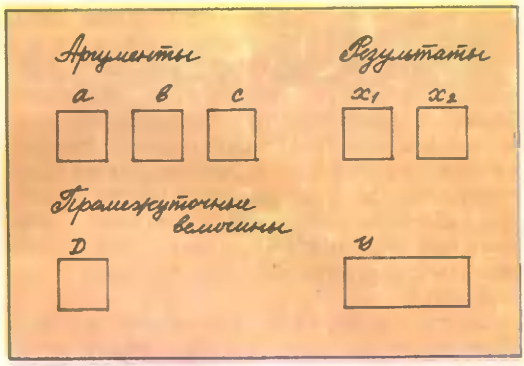
До начала исполнения алгоритма исполнителю сообщаются значения коэффициентов a, b, c. Пусть $a=2$, $b=1$, $c=-6$. Они записываются на доске (рис. 2). Теперь приступаем к выполнению алгоритма. Первая команда ($D:=b^2-4ac$) предписывает записать в рамку с надписью D значение выражения b^2-4ac при данных a, b и c; оно равно 49. Затем надо проверить условие «$D<0$». У нас $D=49$, условие не соблюдается. Следовательно, необходимо выполнить серию команд, идущих в команде ветвления за словом **иначе**. В рамке «y» пишем «есть решения», а в рамки «x_1» и «x_2» записываем корни уравнения, вычисленные по указанным в алгоритме формулам. На этом исполнение алгоритма заканчивается (рис. 3).

При следующем исполнении алгоритма (с другими аргументами) рамки нужно оставить без изменения, а надписи в них стереть.

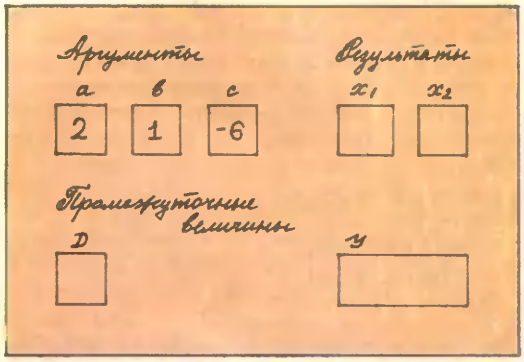
Некоторые рамки могут остаться пустыми. Например, при $D<0$ рамки «x_1» и «x_2» не заполняются. В этом случае говорят, что значения соответствующих величин — неопределенные.

Следующий пример (8.1 из УП) показывает, что значения величин (числа в рамках) могут многократно меняться в ходе исполнения алгоритма.

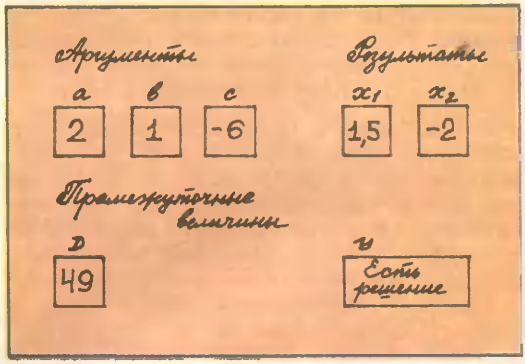
Подготовим доску к его исполнению (рис. 4). Заносим исходные данные ($M=35$, $N=21$), затем исполняем команды $x:=M$; $y:=N$ (числа из рамок «M», «N» переписываем в рамки «x», «y») (рис. 5). Далее выполняется команда повторения



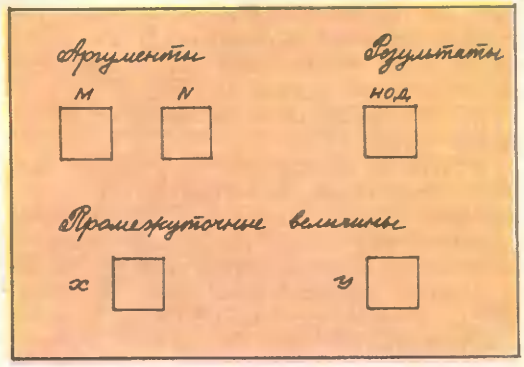
1



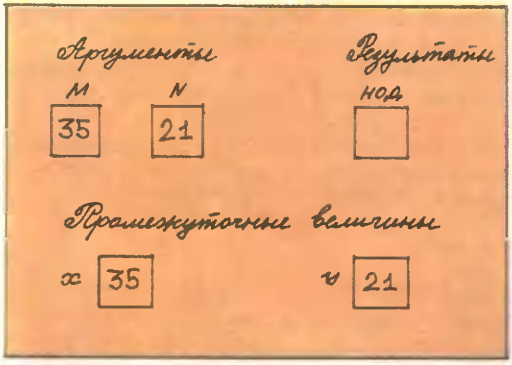
2



3



4



5

```
пока x≠y
нц
    если x>y
      то x:=x—y
      иначе y:=y—x
    все
кц
```

При описании ее исполнения будем рисовать только часть доски, содержащую значения переменных x и y.

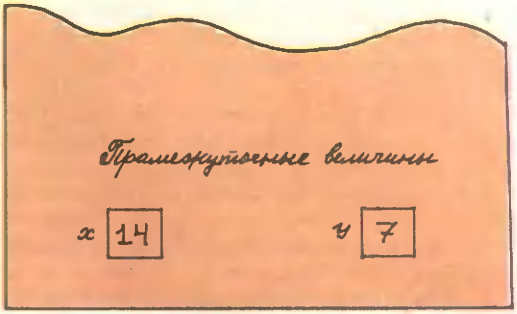
Условие $x \neq y$ соблюдается, выполняется серия команд от **нц** до **кц**, в данном случае одна команда — ветвления. В ней условие $x > y$ соблюдается, поэтому выполняется серия после **то**: $x := x - y$ (рис. 6). И снова $x \neq y$, но теперь $x < y$, и поэтому выполняется присваивание $y := y - x$ (рис. 7). Теперь $x \neq y$, $x > y$; выполняется команда $x := x - y$ (рис. 8), после этого $x = y$. Условие в команде повторения не соблюдается, ее выполнение заканчивается. Остается исполнить последнюю команду алгоритма (НОД:=x) и закончить исполнение алгоритма Евклида (рис. 9).

Как мы видели, в ходе исполнения алгоритма значения промежуточных величин могут меняться. А могут ли меняться значения аргументов и результатов? В УП об этом ничего не сказано. Логичнее всего запретить алгоритму менять значения аргументов и разрешить менять значения результатов. Заметим, что в программе Е-86 («Е-практикум») изменение значений аргументов не рассматривается как ошибка в алгоритме, но может повлечь различные неожиданные и странные явления, если этот алгоритм используется как вспомогательный.

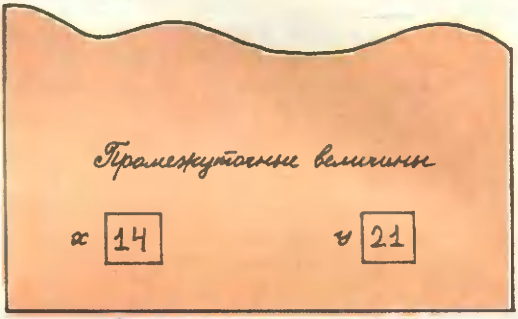
Команда присваивания. Предлагаем несколько упражнений, позволяющих лучше понять механизм ее работы.

1. Значение величины x равнялось 3. Каким оно будет после выполнения команды присваивания $x := 5$?

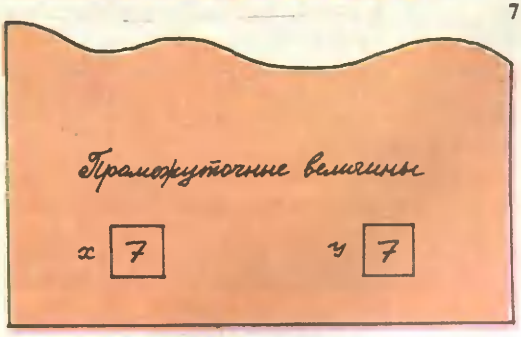
Ответ: 5.



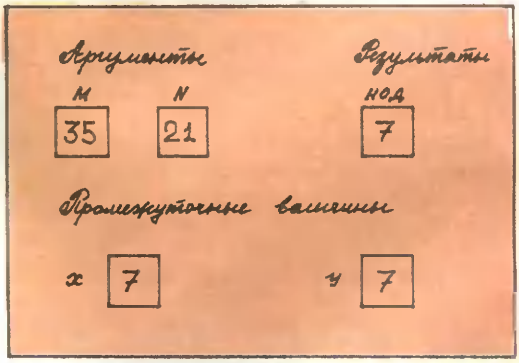
6



7



8



9

2. Значение величины x равнялось 3. Каким оно будет после выполнения команды присваивания $x := x + 5$?

Ответ: 8.

3. Каким было значение величины x, если после выполнения команды присваивания $x:=x+5$ оно стало равным 81?

Ответ: 76.

4. Каким было значение величины x, если после выполнения команды присваивания $x:=6$ оно стало равным 9?

Ответ. Описанная ситуация невозможна. После выполнения команды присваивания $x:=6$ значение величины x может быть равным только 6.

5. Каким могло быть значение величины x, если после выполнения команды присваивания $x:=-x$ оно стало положительным?

Ответ. Значение величины x могло быть равно любому отрицательному числу.

6. Каким могло быть значение величины x, если после выполнения команды $x:=x-2$ оно стало отрицательным?

Ответ. Значение величины x было меньше 2.

7. Какие значения будут иметь величины x и y после выполнения команды $x:=y$, если до этого было $x=4$, $y=6$?

Ответ. $x=6$, $y=6$.

8. Какие значения имели величины x и y до выполнения команды присваивания $x:=y$, если после ее выполнения $x=3$, $y=5$?

Ответ. Описанная ситуация невозможна. После выполнения команды присваивания $x:=y$ значения величин x и y становятся равными.

9. Какие значения имели величины x и y до выполнения команды присваивания $x:=y$, если после ее выполнения $x=3$, $y=3$?

Ответ. Значение величины y было равно 3. Чему было равно значение величины x, установить невозможно, так как в результате выполнения команды присваивания оно утрачено («затерто» новым значением).

10. Какие значения имели величины x и y до выполнения команды присваивания $x:=x+y$, если после ее выполнения $x=2$, $y=3$?

Решение. Величина y не менялась, следовательно, ее значение было равно 3. Значение величины x увеличилось на 3 и стало равно 2. Значит, оно было равным -1.

Ответ: $x=-1$, $y=3$.

11. При составлении алгоритма Васе понадобилось поменять местами значения величин x и y, т. е. сделать так, чтобы новое значение x равнялось старому значению y, а новое значение y равнялось старому значению x. Он написал так:

$x:=y$
$y:=x$

Правильно ли это?

Решение. Нет. После выполнения первой команды старое значение величины x безвозвратно теряется. Например, если до начала исполнения было $x=5$, $y=3$, то после первого присваивания будет $x=3$, $y=3$, и вторая команда ничего не изменит.

12. Как правильно решить предыдущую задачу (разрешается использовать дополнительную переменную величину t).

Решение. Можно написать

$t:=x$
$x:=y$
$y:=t$

После первого присваивания старое значение x будет «запомнено» в t. Затем величина x получит новое значение, равное старому значению величины y. Наконец, величина y получит значение, равное старому значению величины x, сохраненному в t.

Приведем в заключение две более трудные задачи.

13. Решить задачу 12 без использования дополнительной переменной величины.

Ответ. $x:=x+y$; $y:=x-y$; $x:=x-y$.

14. Значения величин x и y равнялись соответственно 7 и 17. Была выполнена последовательность команд присваивания, каждая из которых совпадала с одной из четырех:

$x:=x+y$
$x:=x-y$
$y:=y+x$
$y:=y-x$

Могло ли после этого получиться: а) $x=39$, $y=15$; б) $x=1$, $y=48$?

Указание. Первое невозможно, так как в результате выполнения любой из приведенных команд наибольший общий делитель чисел x и y не меняется. Второе возможно: нужно применить к паре значений (7, 17) последовательность действий алгоритма Евклида, переводящую ее в пару (1, 1), после чего получить пару (1, 48) уже легко.

Примеры алгоритмов работы с величинами

1. Алгоритм сложения дробей.

```
алг сумма (цел числ1, знам1, числ2, знам2, числ, знам)
    арг числ1, знам1, числ2, знам2
    рез числ, знам
нач
    числ:=числ1×знам2+числ2×знам1
    знам:=знам1×знам2
кон
```

После его исполнения (числ/знам) = = (числ1/знам1) + (числ2/знам2).

2. Алгоритм деления с остатком положительных целых чисел.

```
алг деление (цел делимое, делитель, частное, остаток)
```

```
    арг  делимое, делитель
    рез  частное, остаток
нач
    частное:=0
    остаток:=делимое
    пока остаток≥делитель
    нц
      остаток:=остаток—делитель
      частное:=частное+1
    кц
кон
```

На каждом шаге этого алгоритма делимое=делитель×частное+остаток, остаток≥≥0, а после завершения команды повторения остаток меньше делителя.

3. Алгоритм возведения вещественного числа в положительную целую степень.

```
алг  положительная степень (вещ a, цел n,
вещ x)
    арг  a, n
    рез  x
нач  цел k
    k:=1; x:=a
    пока k≠n
    нц
      k:=k+1
      x:=x×a
    кц
кон
```

Здесь поддерживается выполнение соотношения $a^k=x$, для чего выбираются соответствующие начальные значения k и x, а затем при увеличении k соответствующим образом меняется x. В конце выполнения команды повторения входящее в нее условие не соблюдается, т. е. $k=n$ и, следовательно, $x=a^n$. Это и есть результат исполнения алгоритма.

Приведенные примеры (как и примеры в УП) довольно просты. У лучших учеников при их изучении возникает ощущение, что вообще составление алгоритмов — дело нехитрое и не слишком интересное. Чтобы продемонстрировать, что оно может требовать изобретательности, полезно предложить им разобрать пример нетривиального алгоритма, скажем, более хитрого (и более эффективного — требующего меньшего числа действий) алгоритма возведения в целую положительную степень.

```
алг  степень (вещ a, цел n, вещ x)
    арг  a, n
    рез  x
нач  вещ t, цел k
    x:=1; t:=a, k:=n
    пока k≠0
    нц
      если k четно
          то
            k:=k/2; t:=t²
          иначе
            k:=k—1; x:=x×t
      все
    кц
кон
```

Ключом к пониманию этого алгоритма является такое замечание: после любого числа циклов выполняется соотношение $a^n=xt^k$. В самом деле, деление k на 2 компенсируется возведением t в квадрат, а уменьшение k на единицу компенсируется умножением x на t. В конце, когда $k=0$, из указанного соотношения вытекает, что $x=a^n$, что и требовалось.

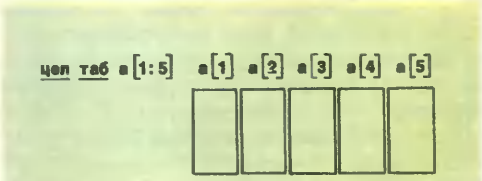
Табличные величины

С помощью команды повторения можно «спрятать» в коротком алгоритме большой объем действий. Табличные величины играют аналогичную роль по отношению к данным: с их помощью можно составить короткий алгоритм, работающий с огромным объемом информации.

Значением табличной величины является не одно число, а целая таблица, заполненная числами. Изображая ее на доске, мы рисуем рамку, разделенную на части (рис. 10). Вообще говоря, в клетках одной таблицы можно хранить совершенно разную по смыслу информацию — нужно только, чтобы типы хранимых величин были одинаковы, но обычно в таблице хранят однородную информацию.

Некоторые вопросы, связанные с табличными величинами, плохо продуманы авторами УП. Прежде всего это относится к таблицам с переменными границами (типа цел таб A[1:n] или вещ таб x[m:n]). Пусть аргументом алгоритма является таблица с переменными границами. Нужно ли перечислять эти границы отдельно в списке аргументов? Могут ли границы быть выражениями или только переменными? Что будет, если сообщенные при исполнении алгоритма значения границ не совпадают с фактическими границами таблицы?

Чтобы не вдаваться в обсуждение этих вопросов, мы предлагаем ограничиться таблицами, границы которых постоянны и заданы явно при их описании. Кроме того,



10

ограничимся линейными таблицами, нумерация в которых начинается с единицы (нижняя граница равна 1). Тогда слова «*i*-й элемент таблицы» можно будет употреблять, не опасаясь двусмысленности: ведь если нижняя граница таблицы не равна 1, то *i*-й по порядку элемент таблицы и элемент с номером (индексом) *i* — разные элементы!

Поиск заданного элемента в таблице. Эта задача, с которой начинается § 6 УП, довольно сложна. Целесообразно вначале рассмотреть более простой ее вариант, когда нужный элемент заведомо имеется в таблице. Если при его разборе в классе возникнут трудности, то от более сложного варианта придется отказаться.

Идею алгоритма можно пояснить следующей аналогией. Пусть где-то на дороге лежит потерянная галоша, которую надо найти. Как это сделать? Идти по дороге и смотреть себе под ноги, пока галоша не обнаружится. Условная запись на алгоритмическом языке будет такой:


```
пока галоши нет под ногами
нц
   шаг вперед
кц
```

Теперь переведем сказанное на язык таблиц. Пусть нам нужно найти в таблице вещ таб ключ [1:1000] элемент, равный заданному числу L (предполагается, что такой элемент в таблице есть). Представим ее себе как дорогу, по которой мы будем идти от начала (1) к концу (1000). Элемент, равный L, и будет искомой галошей. Номер клетки таблицы, в которой мы находимся, обозначим через *k*.

Условие «галоши нет под ногами» переводится на язык таблиц как ключ[*k*]≠L. Команда «шаг вперед» переводится как *k*:=*k*+1. Получается алгоритм

```
алг поиск (вещ таб ключ [1:1000],
вещ L, цел k)
   арг ключ[1:1000], L
   рез k
нач
   k:=1
   пока ключ[k]≠L
   нц
      k:=k+1
   кц
кон
```

После выполнения команды повторения входящее в нее условие (в данном случае ключ[*k*]≠L) не соблюдается, т. е. ключ[*k*]=L; нужный элемент найден.

А что будет, если число L не содержится в таблице? Условие в команде повторения будет соблюдаться, а величина *k* увеличиваться, пока не станет равной 1001. В этот момент при проверке условия произойдет «авария», так как элемента ключ[1001] в таблице нет. Таким образом, если мы хотим предусмотреть и случай безрезультатного поиска, нужно изменить алгоритм. Один из вариантов такого изменения приведен в УП, мы предложим другой.

Прежде всего алгоритм должен иметь возможность сообщить, что искомый элемент в таблице отсутствует; добавим к числу результатов литерную величину «ответ», которая будет равна «найден» или «не найден» в зависимости от эффекта поиска. Введем также целочисленную величину *t*, которая равна числу «просмотренных» элементов таблицы. Между величинами *k*, ответ и *t* будет поддерживаться такая связь:

если L не встречается среди ключ[1], ...ключ[*t*], то ответ=«не найден»;

если встречается среди ключ[1], ...ключ[*t*], то ответ=«найден», ключ[*k*]=L.

Начальные значения величин: *t*=0 (не просмотрено ни одного элемента таблицы), ответ=«не найден».

```
алг поиск (вещ таб ключ[1:1000],
вещ L, цел k, лит ответ)
   арг ключ[1:1000], L
   рез k, ответ
нач цел t
   t:=0; ответ:=«не найден»
   пока t≠1000
   нц
      t:=t+1
      если ключ[t]=L
         то
            k:=t
            ответ:=«найден»
      все
   кц
кон
```

Замечания. 1. После исполнения команды *t*:=*t*+1 значение *t* становится равным номеру первого непросмотренного элемента таблицы.

2. После исполнения команды повторения условие *t*≠1000 не соблюдается, т. е. *t*=1000 (все элементы просмотрены).

3. Если элементы, равные L, встречаются в таблице несколько раз, значение *k* после исполнения алгоритма будет равно номеру наибольшего из этих элементов.

4. Алгоритм можно изменить, если после обнаружения первого элемента, равного L, дальнейший поиск не нужен. Для этого условие команды повторения надо сделать таким:

пока $t \neq 1000$ и ответ=«не найден»

Остальная часть алгоритма остается без изменений. Заодно можно избавиться от дополнительной переменной t: достаточно убрать из текста алгоритма цел t и $k:=t$, а в остальных местах заменить t на k.

Если число L встречается в таблице несколько раз, этот алгоритм находит минимальное k, для которого ключ$[k]$=L.

Поиск наименьшего элемента в таблице. Аргументом этого алгоритма будет вещественная таблица вещ таб $a[1{:}1000]$. Результатом должен быть номер наименьшего элемента в таблице, т. е. такое число l, что $a[l] \leqslant a[1], \ldots, a[1000]$ (заметим сразу же, что таких элементов может быть несколько).

Идея алгоритма состоит в следующем. Пусть мы уже знаем, что наименьшим элементом таблицы $a[1],\ldots,a[n-1]$ является $a[i]$. Добавим к таблице еще один элемент $a[n]$. Что можно сказать о наименьшем элементе таблицы $a[1],\ldots,a[n]$?

Тут возможны два варианта. Если $a[n] < a[i]$, т. е. добавленный элемент меньше минимального из старых элементов, то он будет минимальным в новой таблице. Если же нет ($a[n] \geqslant a[i]$), то $a[i]$ по-прежнему будет минимальным.

Получаем такой алгоритм:

```
алг МИНЭЛЕМЕНТ (вещ таб a[1:1000], цел l)
  арг a
  рез l
нач цел i, n
  i:=1; n:=1
  пока n≠1000
  нц
    n:=n+1
    если a[n]<a[i]
      то
        i:=n
    все
  кц
  l:=i
кон
```

В начале работы алгоритма $n=i=1$ (в таблице из одного элемента этот элемент минимален). После любого числа циклов i является номером элемента, минимального среди $a[1],\ldots,a[n]$. Команда $n:=n+1$ нарушает это соотношение: после ее выполнения можно лишь утверждать, что i — номер элемента, минимального среди $a[1],\ldots, a[n-1]$ (ведь n увеличилось). Следующая за ней команда ветвления восстанавливает нарушенное соотношение. После завершения выполнения команды повторения входящее в нее условие не соблюдается, $n=1000$. Следовательно, $a[i]$ является минимальным элементом всей таблицы. Остается лишь присвоить результату алгоритма (величине l) значение величины i. Можно было бы и не использовать промежуточную величину i, употребляя вместо нее l и исключив команду $l:=l$.

В УП эта задача рассматривается для таблиц с переменными границами. Во избежание путаницы мы заменили переменные границы на постоянные (правда, теперь для использования этого алгоритма как вспомогательного при решении задачи сортировки нам понадобится вернуться к нему и несколько изменить).

Продолжение следует

Я. ЗАЙДЕЛЬМАН
г. Уфа

Язык программирования

Цели изучения языка программирования сформулированы в книге для учителя: «первая — показать школьникам, как конструкции алгоритмического языка могут быть выражены средствами языка программирования (иллюстративная цель); вторая — дать учащимся возможность проработать на ЭВМ... алгоритмы, которые они освоили на уроках (прикладная цель)». Этим определяются ответы на два важнейших вопроса: какой язык изучать и как построить его изучение?

Прикладная цель диктует: изучать надо язык, имеющийся на доступной вычислительной технике; желательно, чтобы по форме и идеям он был как можно ближе к уже изученному алгоритмическому.

Иллюстративная цель определяет методику изучения: язык программирования рассматривается как одна из форм записи алгоритмов, изложение ведется в постоянном сопоставлении с алгоритмическим языком. Необходимо выделить известные по алгоритмическому языку элементы, показать об-

щее в их сущности и различия в оформлении, выяснить особенности, ограничения и дополнительные возможности языка программирования.

Рекомендуем придерживаться следующего плана:

1) основные элементы языка, постоянные и переменные величины, простые типы данных, выражения;

2) общая структура программы, описания, присваивания, ввод и вывод;

3) ветвления и циклы;

4) табличные величины;

5) вспомогательные алгоритмы;

6) дополнительные возможности языка программирования.

По этой схеме можно вести занятия в машинном и безмашинном вариантах, в классе, на факультативе или в кружке, разница лишь в степени подробности изучения.

Темы 1 и 2 дают общее представление о языке программирования. После их изучения можно писать и исполнять на ЭВМ простейшие программы. Особое внимание следует уделить вводу/выводу, организации диалога с ЭВМ, так как это принципиально новое понятие, которого нет в алгоритмическом языке.

Темы 3—5 рассматриваются по аналогии с уже известными конструкциями алгоритмического языка. Главная цель здесь — показать различия в оформлении и единство внутренней сущности.

Тема 6 предназначена для углубленного изучения языка на факультативных и кружковых занятиях.

Там, где есть возможность работать на машине, необходимо рассмотреть еще одну тему: организация исполнения программы на ЭВМ. Здесь обсуждаются вопросы, не имеющие прямого отношения к языку, но необходимые для практической работы: как ввести и откорректировать программу, как запустить ее на исполнение и т. д. Если практические занятия проводятся регулярно, эта тема разбирается сразу же после темы 2, если же они откладываются до экскурсий в конце года, тема разбирается после изучения всего курса.

Прежде чем перейти к анализу конкретных языков программирования, необходимо уяснить отличие языка от его реализации.

Язык — это способ записи алгоритмов, предназначенных для исполнения на ЭВМ.

Реализация — это программа (или комплекс программ), которая переводит текст с языка программирования на язык команд конкретной ЭВМ. В простейшем случае это одна программа — транслятор или интерпретатор.

При создании реализации неизбежны отклонения от стандарта языка. Уточняются некоторые детали, вводятся ограничения, связанные с конкретной ЭВМ, добавляются новые возможности. В результате разные реализации имеют разные возможности, образуются разные *версии* языка.

Так что, прежде чем приступать к урокам, необходимо узнать особенности той версии, которая реализована на доступной технике.

Действующая программа оставляет выбор языка за учителем. Обычно изучаются такие языки, как Бейсик, Фокал, Рапира, Фортран, ПЛ/1, Паскаль, Си. Мы дадим краткие рекомендации по изучению этих языков с расчетом на педагогов, знающих тот или иной язык, а Паскаль изложим более подробно.

Фортран — самый старый (первая версия создана в 1954 г.) язык высокого уровня. Видоизменяясь и приспосабливаясь, он дожил до наших дней, сохранив в себе многочисленные пережитки прошлого. В Фортране плохо развиты алгоритмические конструкции и структуры данных (нет, например, литерных величин), много условностей и нелепых ограничений, поэтому мы не рекомендуем его для изучения.

К сожалению, зачастую другие языки недоступны (обычно это происходит при использовании техники шефствующих предприятий). В этом случае советуем придерживаться следующих рекомендаций (номера здесь и далее соответствуют предложенным выше темам).

1. При изучении типов данных не рассматривайте указатели длины. Используйте только простые типы INTEGER и REAL, соответствующие типам цел и вещ алгоритмического языка.

2. Трудный элемент Фортрана — форматы ввода/вывода. Некоторые реализации (в основном для машин серии СМ) допускают свободный, определяемый самими данными формат «*». Если есть эта возможность, можно отказаться от изучения прочих форматов, если нет (например, в стандартных трансляторах ЕС ЭВМ) — следует пользоваться лишь простейшими типами форматов I, F, X, литерал.

3. Один из главных недостатков Фортрана — отсутствие алгоритмических конструкций ветвления и цикла пока. Они моделируются с помощью условного оператора (IF) и оператора перехода (GOTO) (табл. 1).

Следует использовать только логический IF, не упоминая о существовании арифметического (с тремя метками).

GOTO должен рассматриваться как средство моделирования алгоритмических конст-

Таблица 1

Перевод алгоритмических конструкций в операторы Фортрана

если условие		IF (условие)	GOTO 1
			GOTO 2
то	1		CONTINUE
серия 1		серия 1	
иначе	С	иначе	GOTO 3
	2		CONTINUE
серия 2		серия 2	
все	3		CONTINUE
пока условие	1	IF (условие)	GOTO 2
			GOTO 3
нц	2		CONTINUE
серия		серия	
кц	3	CONTINUE	GOTO 1

рукций и использоваться только в этих целях.

4. Фортран предлагает слишком много способов для описания массивов (таблиц). Следует выбрать один и использовать только его. Лучше, если это будут операторы REAL и INTEGER, так как они позволяют создать единообразную систему описания величин, максимально приближенную к алгоритмическому языку.

5. В пособиях по Фортрану обычно приводятся детальные правила, определяющие согласования формальных и фактических параметров. Не нужно вдаваться в эти подробности и использовать экзотические возможности типа параметров-меток. Достаточно ограничиться естественным согласованием, примерно соответствующим алгоритмическому языку. Следует подчеркнуть, что за соответствием формальных и фактических параметров по их числу, типу и виду (аргументы и результаты) должен следить программист. Машина соответствия не проверяет, а его нарушение может повлечь тяжкие последствия.

6. В Фортране практически нет принципиальных возможностей, отсутствующих в алгоритмическом языке. Для углубленного изучения языка рекомендуем темы:

1) обработка литерных значений с помощью величин типа LOGICAL*1 (или BYTE, если этот тип есть в реализации);

2) моделирование сложных структур данных (стеки, очереди, списки) с помощью массивов;

3) система ввода-вывода, сложные форматы, работа с двоичными файлами.

Фортран-77. По сравнению с обычным Фортраном он имеет много достоинств. Это в первую очередь литерные величины (тип CHARACTER), полноценное ветвление (IF—THEN—ELSE—ENDIF), богатые возможности ввода-вывода, включающие, в частности, свободный формат. В то же время в нем нет цикла пока, сохранились операторы-пережитки (арифметический IF, присваиваемый GOTO, ASSIGN), не всегда логичен синтаксис.

В схеме изучения отличия от Фортрана минимальны: добавляется тип CHARACTER и все связанные с ним возможности, отпадает необходимость моделирования ветвления с помощью GOTO. В остальном же приведенные выше рекомендации сохраняют силу.

ПЛ/1 создан в начале 60-х гг. В него включены практически все возможности наиболее популярных в то время языков программирования (Алгол, Фортран, Кобол) и многое сверх того. В результате попытки объять необъятное родился язык-монстр, всех богатств которого не знает, вероятно, ни один программист. Однако «минимальный» ПЛ/1, включающий только самые необходимые средства, достаточно прост, удобен в работе и легок в изучении.

Тем, кто проводит практические занятия на ЕС ЭВМ в пакетном режиме и должен выбрать один из стандартных языков ЕС ЭВМ, советуем остановиться на ПЛ/1.

1. Один из самых сложных элементов ПЛ/1 — типы. Их насчитывается несколько сот. Определения типов ориентированы не на человеческое, а на машинное представление, отсюда такие параметры, как система счисления, форма представления, точность. Не следует изучать все это многообразие. Достаточно выделить несколько типов, примерно соответствующих имеющимся в алгоритмическом языке, и пользоваться только ими (табл. 2).

2. Для ввода-вывода в простых учебных программах вполне достаточно режимов LIST и DATA или даже только одного из них.

3. Операторы IF—THEN—ELSE и WHILE

Таблица 2

Соответствие типов алгоритмического языка и ПЛ/1

цел	BINARY	FIXED	(15)
вещ	BINARY	FLOAT	(21)
лит	CHARACTER	(N)	VARYING

(N — максимально допустимая длина литерной величины)

полностью соответствуют конструкциям если и пока алгоритмического языка. Единственная сложность — необходимость оформления серии команд в виде составного оператора (группа DO). Этот аспект будет подробно рассмотрен при анализе Паскаля.

4. Советуем рассмотреть дополнительные возможности работы с массивами, предоставляемые ПЛ/1: операции над массивами, сечения, функции ALL и ANY. Они достаточно просты и существенно увеличивают изобразительные возможности языка, облегчают программирование.

5. В ПЛ/1 различаются внешние и внутренние процедуры. Рекомендуем ограничиться внешними, так как они ближе к вспомогательным алгоритмам алгоритмического языка, при их использовании не возникает сложностей, связанных с понятием глобальной переменной. Однако при использовании внешних процедур изобилие типов в ПЛ/1 приводит к труднонаходимым ошибкам согласования параметров. Чтобы избежать их, можно требовать обязательного описания процедур с помощью оператора DECLARE ENTRY.

6. Из дополнительных возможностей ПЛ/1 в первую очередь стоит рассмотреть:

1) динамическое распределение памяти, управляемые (CONTROLLED) переменные;
2) структуры и операции над ними;
3) систему ввода-вывода.

Рапира, созданная специально для обучения школьников, обладает многими достоинствами. К сожалению, этот язык не получил широкого распространения, поэтому работать с ним могут только школы, имеющие ЭВМ «Агат».

Нам не приходилось иметь дело с «живой» Рапирой (в Уфе больше миллиона жителей, но нет ни одного «Агата»!), к тому же ее основы достаточно подробно изложены в пробном учебном пособии и книге для учителя, поэтому подробных рекомендаций по ней не будет. Остановимся только на двух моментах, которые представляются достаточно важными.

Основное отличие Рапиры от алгоритмического языка — иная трактовка понятия типа величины. Чтобы уберечь учеников от путаницы, необходимо явно сформулировать и объяснить эти различия. В алгоритмическом языке тип связывается с именем величины. Он определяет множество допустимых значений величины и множество операций, в которых она может участвовать. В Рапире переменная величина не имеет фиксированного типа и может принимать любые допустимые значения. Но это не означает, что понятие типа вообще отсутствует. Тип величины существует, но не задается при ее описании, а связывается с текущим значением и определяет множество допустимых операций.

Рапира разработана Г. А. Звенигородским и описана им в серии статей в журнале «Квант» (1980 г.) и в книге «Первые уроки программирования». Однако учитель, обратившийся к этим источникам за дополнительной информацией, столкнется с неожиданным фактом. Рапира, описанная в пробном учебном пособии и книге для учителя, сохраняет все идеи Рапиры Г. А. Звенигородского, но имеет много синтаксических отличий (заменены служебные слова, различаются правила записи присваивания и некоторых других конструкций). Какой же вариант считать правильным? Очевидно, тот, который соответствует имеющейся реализации. Желательно разобраться в этом до начала занятий по данной теме.

Бейсик. Об опасности Бейсика уже много писалось. Его изучение нельзя оправдать даже в том случае, когда он является единственным языком, доступным на имеющейся технике. В такой ситуации лучше вообще отказаться от проведения практических занятий.

Сторонники лозунга «Лучше что-то, чем ничего» забывают, что переучивать труднее, чем учить.

Мы не станем давать никаких рекомендаций по изучению Бейсика, чтобы не создавать иллюзий, что при их соблюдении, при введении некоторых ограничений он не так уж и вреден.

Бейсик действует на мозг подобно алкоголю и наркотикам, нанося непоправимый вред даже в малых дозах.

Фокал. Исчерпывающая характеристика этого языка укладывается в два слова: хуже Бейсика.

Си — язык профессионалов, язык системного программирования. К сожалению, его основные достоинства (лаконичность, богатый набор операций, стройная система использования указателей) оборачиваются недостатками при попытке обучать ему школьников. Поэтому мы не рекомендуем изучать Си на уроках информатики. Тем, кто все же решится сделать это, дадим несколько советов.

Не нужно рассматривать все экзотические особенности языка. Это особенно касается операций. Школьникам будет вполне достаточно привычных операций над привычными типами.

Не стоит вводить присваивание как операцию, входящую в состав сложных выражений. Лучше ограничиться привычной по алгоритмическому языку системой команд.

Особо стоит вопрос об указателях. Их существование можно было бы вообще «утаить» от учеников, если бы не одно обстоятельство: без указателей в Си нельзя реализовать параметры-результаты. Можно попытаться ввести понятие указателя, но он должен рассматриваться только как средство моделирования параметров-результатов.

Паскаль

На примере Паскаля мы подробно рассмотрим предлагаемую схему изучения языка программирования. Такой выбор продиктован следующими соображениями.

Паскаль очень близок к алгоритмическому языку, но в то же время не тождествен ему. Это обеспечивает достижение иллюстративной цели изучения языка программирования.

Паскаль широко распространен. Он реализован на большинстве мини- и микро-ЭВМ, есть реализация и для ЕС ЭВМ. Это упрощает достижение практической цели изучения.

Паскаль богат дополнительными (по сравнению с алгоритмическим языком) возможностями, что позволяет углубленно изучить предмет, обеспечивает естественный переход от обязательного минимума к дополнительным кружковым занятиям, к самостоятельному изучению программирования сильными учениками.

Приведенное ниже описание лишь основы Паскаля. Мы постарались ограничиться минимумом, необходимым для проведения уроков, и продемонстрировать методику перехода от алгоритмического языка к языку программирования. Превышающая необходимый минимум информация, предназначенная для факультатива, выделена мелким шрифтом.

1. Основные элементы языка. *1.1. Числа.* В Паскале различаются целые и вещественные числа. Целые записываются по обычным правилам, принятым в математике: в виде последовательности цифр, перед которой может стоять знак + или —, например 5, —21, 1987.

Вещественные числа могут записываться в форме с фиксированной или плавающей точкой. Запись с фиксированной точкой соответствует обычной математической записи десятичных дробей, только десятичная запятая заменяется точкой: 0.5, 3.14, —27.82.

Запись с плавающей точкой соответствует стандартной форме числа. Для отделения мантиссы от порядка используется символ E: 1E2 (=100), 1.6E—19 ($=1.6\cdot10^{-19}$).

Множество допустимых вещественных чисел определяется минимальным и максимальным (по модулю) значениями и точностью представления (количеством значащих цифр). Все эти параметры зависят от реализации. Дополнительно в множество вещественных чисел входит ноль, хотя он и меньше допустимого минимума.

Целые числа допускаются в диапазоне от MININT до MAXINT. MININT и MAXINT — константы, зависящие от реализации; эти имена можно употреблять в программе как обычные числа. В некоторых реализациях MININT не определено, тогда считается, что MININT=—MAXINT; на ПЭВМ обычно MININT=—32768, MAXINT=32767.

1.2. Литерные постоянные (строки) — текст, ограниченный апострофами: 'A', '16', 'Паскаль'.

Пустые строки не допускаются, максимальная длина ограничивается шириной текста программы: перенос литерной константы на новую строку не разрешается.

1.3. Имена используются для обозначения различных объектов, в первую очередь переменных.

Имя — последовательность букв и цифр, начинающаяся с буквы. В стандартном Паскале имена различаются только по первым 8 символам. Пробелы в имени не допускаются. Но что считать буквой? Согласно стандарту Паскаля, буква — это одна из заглавных или строчных латинских букв, однако на отечественных дисплеях и АЦПУ строчные латинские буквы чаще всего заменены заглавными русскими. Это дает возможность использовать в именах 26 русских букв (кроме Ч, Ш, Щ, Э, Ю, не имеющих латинской пары).

Рекомендуем придерживаться следующего правила: на теоретических занятиях разрешается использовать имена любой длины с любыми русскими буквами — такой подход существенно облегчает написание и понимание программ; при выходе на ЭВМ необходимо учесть ограничения конкретной реализации и внести в программы соответствующие коррективы.

1.4. Переменные. Понятия переменной в Паскале и алгоритмическом языке совпадают. Различаются допустимые типы переменных.

В Паскале программист имеет возмож-

ность описывать свои собственные типы. Конечно, некоторые типы определены изначально, они называются предопределенными, или стандартными. К ним относятся REAL, INTEGER, CHAR, BOOLEAN.

REAL соответствует типу вещ алгоритмического языка. Множество значений совпадает с множеством допустимых вещественных чисел.

INTEGER соответствует типу цел алгоритмического языка. Множество значений совпадает с множеством допустимых целых чисел.

CHAR — литерный. Значение переменной этого типа — один символ текста, в отличие от типа лит алгоритмического языка, для которого значение — упорядоченный набор символов. Множество допустимых значений переменной типа CHAR включает все символы, имеющиеся на ЭВМ, в том числе не входящие в алфавит Паскаля и даже не отображаемые на экране дисплея.

BOOLEAN — булев, или логический. Множество допустимых значений этого типа включает всего две величины: TRUE (истина) и FALSE (ложь). В алгоритмическом языке этот тип не имеет явного аналога, но неявно используется очень широко — условия в командах ветвления и повторения есть не что иное, как выражения с результатом логического типа.

1.5. Выражения. Для стандартных типов определены следующие операции.

Для типов REAL и INTEGER определены операции: унарный минус (перемена знака), сложение (+), вычитание (—), умножение (*) и деление (/). Результат деления всегда имеет тип REAL, результат остальных операций имеет тип INTEGER, если участвуют только величины типа INTEGER, и тип REAL в противном случае.

Для типа INTEGER есть две дополнительные операции: целое деление (DIV) и остаток от деления (MOD). Поясним их на примере:

7/4=1.75;
7DIV4=1;
7MOD4=3.

Для типа BOOLEAN определены операции NOT (не), AND (и), OR (или), соответствующие операциям алгоритмического языка.

Для типа CHAR специальных операций не предусмотрено.

Для всех типов определены операции сравнения: равно (=), не равно (<>), меньше (<), больше (>), меньше-равно (<=), больше-равно (>=). Они дают результат типа BOOLEAN; для типа CHAR сравниваются порядковые номера символов, которые в общем случае зависят от реализации. Латинские буквы и цифры всегда упорядочены ('A'<'B'<...<'Z', 'O'<'1'<...<'9'), но для русских букв это не так, поэтому сравнением литерных величин следует пользоваться осторожно.

Выражения составляются по общепринятым математическим правилам.

Важное значение имеет приоритет операций, определяющий порядок действий при вычислении выражения. В Паскале принято следующее распределение приоритетов:

1) унитарные операции (NOT, —);
2) операции типа умножения (*, /, DIV, MOD, AND);
3) операции типа сложения (+, —, OR);
4) операции сравнения (=, <>, <, >, <=, >=).

При вычислении выражений выполняются влева направо операции первого приоритета, затем второго и т. д. Изменить порядок действий можно с помощью скобок.

Обратите внимание, что приоритет логических операций выше, чем у операций сравнения. Отсюда вытекает необходимость дополнительных скобок в составных условиях. Например, условие A<B<C нельзя записать как A<B AND B<C: в таком случае первой будет выполняться операция B AND B, что приведет к ошибке. Правильная запись (A<B) AND (B<C).

В выражения можно включать обращения к функциям, набор которых зависит от реализации. Некоторые функции, включенные в стандарт Паскаля, приведены в табл. 3.

Обратите внимание, что в Паскале нет операции возведения в степень. Для ее выполнения используется тождество

$$a^b = e^{b \ln a}$$

Таким образом, A^B можно записать как EXP(B*LN(A)).

1.6. Комментарии. В программу на Паскале можно вставлять комментарии — пояснения, предназначенные для человека. При исполнении на ЭВМ комментарии пропускаются (не анализируются машиной и не влияют на ход исполнения программы). Их задача — помочь читающему программу человеку понять ее назначение и ход работы.

Комментарии пишутся в фигурных скобках. Поскольку на отечественных дисплеях и АЦПУ фигурных скобок, как правило, нет, их можно заменять на скобки со звездочкой:

(* комментарий *)

Многие реализации рассматривают русскую букву Щ как правую фигурную скобку. В этом случае использовать букву Щ внутри комментария нельзя!

2. Общая структура программы. Программа на Паскале начинается с заголовка, за ним следуют описания и наконец тело — исполняемая часть программы.

2.1. Заголовок имеет вид

PROGRAM имя (параметры);

Таблица 3

Стандартные функции Паскаля

Функция	Тип аргумента	Тип результата	Пояснение
ABS(X)	INTEGER, REAL	INTEGER, REAL	Абсолютная величина
SQR(X)	—»—	—»—	x^2
SQRT(X)	—»—	REAL	$\sqrt{x}$
EXP(X)	—»—	—»—	e^x
LN(X)	—»—	—»—	$\ln x$
SIN(X)	—»—	—»—	$\sin x$
COS(x)	—»—	—»—	$\cos x$
ARCTAN(X)	—»—	—»—	$\operatorname{arctg} x$
ODD (X)	INTEGER	BOOLEAN	TRUE, если x нечетно
TRUNC (X)	REAL	INTEGER	$[x]$, целая часть
ROUND(X)	—»—	—»—	Округление до ближайшего целого
ORD(X)	CHAR	—»—	Порядковый номер символа x
CHR(X)	INTEGER	CHAR	Символ с номером x

Здесь PROGRAM — служебное слово; имя — имя программы, образованное по общему правилу создания имен; параметры — перечисленные через запятую файлы, используемые в программе.

Для большинства простейших диалоговых программ список параметров имеет вид (INPUT, OUTPUT). Это означает, что для ввода и вывода используются стандартные файлы INPUT и OUTPUT, обычно связанные с клавиатурой и экраном дисплея.

Во многих версиях указывать параметры необязательно; заголовок в этом случае имеет вид

PROGRAM имя;

В некоторых версиях заголовок вообще необязателен, но мы рекомендуем всегда начинать программу с заголовка. Это улучшает оформление и позволяет дать программе содержательное имя.

2.2. Описания. С понятием описания мы знакомы по алгоритмическому языку, где описывались переменные. В Паскале кроме переменных описываются и некоторые другие объекты.

Описания делятся на разделы, которые должны располагаться в следующем порядке:

1) раздел меток (упомянут только для полноты изложения, далее рассматриваться не будет);

2) раздел констант;

3) раздел типов (будет рассмотрен в п. 4);

4) раздел переменных;

5) раздел процедур (будет рассмотрен в п. 5).

Во многих версиях порядок разделов может быть произвольным, одноименные разделы могут встречаться по нескольку раз. Такая свобода очень удобна при написании больших программ, так как позволяет группировать описания объектов не по формальным, а по сущностным признакам.

2.2.1. Описание переменных. Раздел переменных имеет вид

```
VAR
  имя_1, имя_2, ..., имя_N: тип;
  имя, ... : тип;
  ...
```

VAR — служебное слово, сокращенное от английского VARIABLE — переменная.

Как и в алгоритмическом языке, можно указать общий тип для группы имен. Количество групп и количество имен в каждой группе не ограничены. Пример:

```
VAR
  X, Y : REAL;
  I : INTEGER;
  БУКВА : CHAR;
```

Переменные одного типа необязательно собирать в одну группу. Например, следующие два описания равносильны:

```
VAR
  X, Y : REAL;
```

```
VAR
  X : REAL;
  Y : REAL;
```

2.2.2. Описание констант относится к дополнительным (по сравнению с алгоритмическим языком) возможностям Паскаля, но мы включили его в общий обзор, так как этот элемент очень полезен и не слишком сложен.

В алгоритмическом языке используются постоянные и переменные величины. Переменные имеют имя и значение, которое может изменяться, постоянные — только значение, указываемое непосредственно в тексте алгоритма.

В Паскале имеется возможность давать имена постоянным величинам (константам)

и использовать их в тексте программ. Описание таких величин образует раздел констант, который имеет вид

```
CONST
  имя_1 = значение_1;
  имя_2 = значение_2;
  ...
  имя_N = значение_N;
```

Количество описаний не ограничивается. Пример:

```
CONST
  E=1.6E—19; (* кулон, заряд электрона*)
  СТРАНИЦА=60; (*количество строк на листе бумаги*)
```

Зачем нужны именованные константы?

Во-первых, они облегчают чтение и понимание программы. «Магические числа» в ее тексте часто вызывают недоумение, человек не сразу понимает их смысл. Замена числа содержательным именем или общепринятым обозначением снимает эту проблему.

Во-вторых, константы облегчают модификацию программ. Программистский фольклор утверждает, что константа — это переменная, которая меняется реже других. Например, в программе, решающей физическую задачу, может потребоваться провести вычисления в другой системе единиц. В этом случае физические постоянные изменяют численные значения, но сохраняют свой физический смысл. Если значения включены в текст программы, ее исправление — утомительная кропотливая работа, часто порождающая ошибки. При использовании именованных констант исправления сводятся к замене соответствующих значений в единственном месте.

2.3. Тело программы имеет вид, аналогичный телу алгоритма:

```
BEGIN        нач
  группа          серия
  операторов      команд
END.         кон
```

Точка в конце обязательна. Между операторами, входящими в группу, ставится точка с запятой.

2.3.1. Присваивание. Общий вид оператора присваивания:

```
имя := выражение
```

По смыслу и форме это полностью соответствует присваиванию в алгоритмическом языке.

Имя и выражение должны иметь одинаковый тип; единственное исключение — допускается присваивание целого значения вещественной переменной (но не наоборот. Чтобы присвоить целой переменной вещественное значение, его нужно привести к целому с помощью функций TRUNG или ROUND).

2.3.2. Ввод и вывод. В языках программирования, в том числе в Паскале, проблема связи программы с внешним миром решается с помощью принципиально нового, отсутствующего в алгоритмическом языке механизма — ввода-вывода.

Операция вывода обеспечивает передачу информации от ЭВМ человеку. Простейшие формы команды вывода в Паскале:

```
WRITE (список выражений)
```

или

```
WRITELN (список выражений)
```

При выполнении любой из этих команд (строго говоря, это не команды, а обращения к стандартным процедурам) на экране дисплея последовательно появляются значения всех выражений, входящих в список. Он может включать любые выражения, перечисленные через запятую; особенно часто используют переменные и текстовые константы.

Различие WRITE и WRITELN в том, что WRITELN обеспечивает переход на новую строку (после своего выполнения).

Операция ввода обеспечивает передачу информации от человека к ЭВМ. Простейшие формы команды ввода в Паскале:

```
READ (список переменных)
READLN (список переменных)
```

При выполнении этих команд ЭВМ приостанавливает работу программы. Человек должен ввести с клавиатуры дисплея значения переменных, входящих в список. Числовые значения вводятся через запятую, литерные — без разделителей (проблемы разделения не возникает, так как длина переменной типа CHAR всегда равна 1). Ввод завершается нажатием специальной клавиши. Введенные значения присваиваются соответствующим переменным, после чего выполнение программы продолжается.

Различие READ и READLN в том, что после выполнения READLN происходит переход на новую строку. Например, при выполнении серии

```
READ (A);
READLN (B);
READ (C)
```

значения A и B должны быть введены в одной строке, а значение C — в другой.

При написании диалоговых программ необходимо предварять каждую команду ввода выводом сообщения, поясняющего, что именно следует ввести.

Теперь можно писать простейшие программы на Паскале. Построим алгоритм и программу для решения такой задачи: определить длину окружности и площадь круга заданного радиуса.

```
алг КРУГ (вещ R, L, S)
  арг R
```

```
   рез L, S
нач вещ ПИ
   ПИ :=3.14
   L :=2*ПИ*R
   S := ПИ*R**2
кон
```

```
PROGRAM КРУГ;
   CONST ПИ=3.14;
   VAR R, L, S : REAL;
BEGIN
   WRITE ("ВВЕДИТЕ РАДИУС
                              КРУГА : ");
   READLN(R);
   L :=2*ПИ*R;
   S :=ПИ*SQR(R);
   WRITELN ("ДЛИНА
                        ОКРУЖНОСТИ", L);
   WRITELN ("ПЛОЩАДЬ
                              КРУГА", S);
END.
```

Операторы ввода-вывода соответствуют аргументам и результатам алгоритмического языка. Программа начинается с ввода аргументов и заканчивается выводом результатов.

Работа школьников в пакетном режиме нежелательна, но реальные условия таковы, что многие школы вынуждены проводить занятия на ЕС ЭВМ, а некоторые даже общаться с ВЦ по почте. Учитывая это, коротко рассмотрим основные особенности ввода-вывода в этом режиме.

Программа, работающая в пакетном режиме, обычно вводит данные с перфокарт и выводит результаты на АЦПУ.

Ввод с перфокарт означает, что данные должны быть подготовлены заранее. Подсказка человеку во время работы программы теряет смысл. Человек должен знать запросы программы до того, как запускать ее.

Однако при анализе результатов работы программы полезно видеть, какие значения были заданы аргументам. В диалоговом режиме на экране сохраняются значения, введенные человеком. В пакетном этого нет, поэтому поясняющие сообщения перед вводом рекомендуется заменять на распечатку введенных значений сразу после него. Рассмотрим пример.

```
PROGRAM КРУГ (INPUT,
                              OUTPUT);
   CONST ПИ=3.14;
   VAR R, L, S : REAL
BEGIN
   READLN (R);
   WRITELN ("ВВЕДЕННЫЙ
                        РАДИУС", R);
   L:=2*ПИ*R;
   S:=ПИ*SQR(R);
   WRITELN ("ДЛИНА ОКРУЖНОСТИ", L);
   WRITELN ("ПЛОЩАДЬ КРУГА" , S);
END.
```

3. Ветвления и циклы. *3.1. Ветвление и цикл* пока. В Паскале имеются условный оператор IF и оператор цикла WHILE, аналогичные конструкциям если и пока алгоритмического языка. Их синтаксис:

```
IF      условие
        THEN оператор
        ELSE оператор

WHILE условие DO
        оператор
```

Условие — произвольное выражение типа BOOLEAN (т. е. принимающее значение *истина* либо *ложь*, что соответствует понятию условия алгоритмического языка). Исполняются эти операторы так же, как хорошо знакомые нам если и пока.

Однако есть и отличие. Там, где алгоритмический язык допускает серию команд, в Паскале должен стоять единственный оператор. Как следствие этого отпадает необходимость в служебных словах, явно закрывающих конструкцию (типа все и кц).

Что же делать, если в ветвление или цикл необходимо включить именно серию? Для этого в Паскале предусмотрен составной оператор. Его синтаксис:

```
BEGIN
оператор;
оператор;
. . .
оператор
END
```

Группа операторов, заключенных между служебными словами BEGIN и END (эти слова иногда называют операторными скобками), рассматривается как единственный оператор и может включаться в IF и WHILE. Использование составных операторов обеспечивает полное совпадение возможностей ветвлений и циклов в алгоритмическом языке и Паскале.

Однако ученики, привыкшие к более простому синтаксису алгоритмического языка, испытывают некоторые трудности при использовании слов BEGIN и END и расстановке точек с запятой. BEGIN и END — это скобки, необходимые при включении в алгоритмическую конструкцию группы операторов, точка с запятой — разделитель между отдельными операторами, но не их составная часть. Поэтому точка с запятой не ставится перед END и ELSE — в этих местах нет границы операторов. Сознательное применение данных правил требует глубокого усвоения паскалевской трактовки понятий «оператор» и «составной оператор», тонкого понимания структуры Паскаль-программ. Нельзя рассчитывать, что этого достигнет каждый ученик. Поэтому мы предложим формальные

правила, пользуясь которыми легко перевести на Паскаль любой алгоритм.

При переводе алгоритмических конструкций можно механически заменять служебные слова в соответствии с табл. 4.

При этом получается такое соответствие конструкций:

Таблица 4

если	IF
то	THEN BEGIN
иначе	END ELSE BEGIN
все	END
пока	WHILE
нц	DO BEGIN
кц	END

```
если  УСЛОВИЕ                IF  УСЛОВИЕ  THEN
                                  BEGIN
  то  СЕРИЯ-1                     СЕРИЯ-1
                                  END
иначе                         ELSE
                                  BEGIN
СЕРИЯ-2                           СЕРИЯ-2
все                           END

пока  УСЛОВИЕ                 WHILE  УСЛОВИЕ  DO
нц                                BEGIN
    СЕРИЯ                         СЕРИЯ
кц                                END
```


Данная замена автоматически обеспечивает использование составных операторов. Если между BEGIN и END окажется единственный оператор, операторные скобки можно опустить.

Точка с запятой ставится:

1) после заголовка программы — всегда;
2) после каждого описания — всегда;
3) после присваивания, обращения к процедуре (в т. ч. ввода-вывода), слова END — в том случае, если после «;» не окажется одно из слов END, ELSE, WHILE.

Во всех остальных случаях «;» *не ставится*.

В качестве примера приведем перевод на Паскаль двух алгоритмов из учебного пособия.

```
PROGRAM квур;
  VAR
  A, B, C: REAL;
  X1, X2 : REAL;
  D : REAL;
BEGIN
  WRITE ("ВВЕДИТЕ КОЭФФИЦИЕНТЫ КВАДРАТНОГО УРАВНЕНИЯ:");
  READLN (A, B, C);
  D:=SQR(B)-4*A*C;
  IF D<0 THEN
     BEGIN
     WRITELN ("РЕШЕНИЙ НЕТ")
     END
  ELSE
     BEGIN
     X1:=(-B-SQRT(D))/(2*A);
     X2:=(-B+SQRT(D))/(2*A);
     WRITELN ("КОРНИ УРАВНЕНИЯ: ", X1, X2)
     END
END.
```

Обратите внимание, что литерная переменная Y становится ненужной: информационное сообщение выводится непосредственно на экран.

```
PROGRAM нод;
  VAR
  M, N : INTEGER;
  X, Y : INTEGER;
BEGIN
  WRITE ("ВВЕДИТЕ ДВА НАТУРАЛЬНЫХ ЧИСЛА:");
  READLN (M, N);
  X:=M; Y:=N;
  WHILE X<>Y DO
     BEGIN
     IF X>Y THEN
```

```
      BEGIN
      X:=X—Y
      END
   ELSE
      BEGIN
      Y:=Y—X
      END
   END;
 WRITELN ('НАИБОЛЬШИЙ ОБЩИЙ ДЕЛИТЕЛЬ ЧИСЕЛ', M, 'И', N, 'РАВЕН', X)
END.
```

Этот пример демонстрирует, что при механической замене служебных слов алгоритмического языка программа становится громоздкой, в ней много лишних операторных скобок, но зато синтаксическая правильность гарантируется.

Рассмотрим еще две конструкции Паскаля. Их включение в ученический минимум необязательно, но учитель должен знать их.

3.2. Цикл до. Синтаксис:

```
REPEAT
серия
UNTIL условие
```

(обратите внимание: цикл до имеет собственное закрывающее слово и не нуждается в составном операторе).

Механизм работы: выполняется серия, затем проверяется условие; если оно ложно, снова выполняется серия, проверяется условие и т. д. Если условие истинно, цикл завершается.

Основные отличия от цикла пока:

1) тело цикла до гарантированно исполняется хотя бы один раз, так как проверка условия происходит в конце, а не в начале цикла;

2) вместо условия продолжения цикла указывается условие его завершения.

3.3. Выбор (CASE). Синтаксис:

```
CASE выражение OF
     список_1 : оператор_1;
     список_2 : оператор_2;
        . . .
     список_N : оператор_N
END
```

Выражение может быть любого простого типа, кроме REAL. Списки состоят из перечисленных через запятую констант соответствующего типа.

Механизм работы: вычисляется значение выражения; если это значение включено в список_1, выполняется оператор_1, если в список_2,— оператор_2 и т. д. После выполнения одного оператора остальные списки не проверяются.

Допускаются любые, в том числе составные, операторы.

Эта конструкция аналогична команде выбора алгоритмического языка, но выбор производится не по произвольному условию, а по значению выражения, что ограничивает возможности конструкции.

4. Табличные величины (массивы). *4.1. Описание массивов.* Присмотримся к привычному описанию таблиц в алгоритмическом языке.

```
Общий вид:  ТИП  таб  ИМЯ  [НГ : ВГ]
Пример:     вещ  таб  A    [1 : 100]
```

Описание включает 4 основных элемента:

1) служебное слово таб, указывающее, что описывается таблица;

2) имя таблицы;

3) тип элементов, составляющих таблицу;

4) нижнюю и верхнюю границы индексов (номеров) элементов.

Описание массивов в Паскале выглядит несколько иначе, но содержит в точности те же самые элементы.

Общий вид: имя : ARRAY [нг .. вг] OF тип;

Пример: A : ARRAY [1 .. 100] OF REAL;

Описание включает служебное слово ARRAY (массив), имя таблицы, тип элементов, границы индексов. Слово OF смысловой нагрузки не несет, его присутствия потребовала грамматика английского языка, а не языка программирования.

Обратите внимание, что двоеточие, разделяющее нижнюю и верхнюю границы, «положено набок». Две точки — принятый в Паскале знак интервала.

Границы индексов могут быть только константами (простыми или именованными). Это серьезный недостаток. В алгоритмическом языке границы можно было задавать параметрами-аргументами, что позволяло обрабатывать таблицы, размер которых заранее неизвестен. В Паскале такой возможности нет.

Тип индексов в алгоритмическом языке может быть только целым. В Паскале допускаются индексы любого перечислимого типа (в частности, любого простого типа, кроме REAL).

Приведенное описание массивов включается в раздел переменных. Есть еще один способ описания, связанный с использованием раздела типов.

Описание типов — это средство, с помощью которого программист может конструировать свои типы и давать им имена. Раздел типов имеет вид

```
TYPE
   имя_типа=описание_типа;
   имя_типа=описание_типа;
   . . .
```

Пример.

```
TYPE
   ТАБЛИЦА=ARRAY [1..100] OF REAL;
VAR
   A : ТАБЛИЦА;
```

Здесь введен новый тип, называющийся ТАБЛИЦА. Переменные этого типа, одна из которых — А, представляют собой массивы вещественных элементов с нумерацией от 1 до 100.

Прямоугольные таблицы рассматриваются как массивы массивов и описываются соответственно:

имя : ARRAY [нг1 .. вг1] OF ARRAY [нг2 .. вг 2] OF тип;

или

имя : ARRAY [нг1 .. вг1, нг2 .. вг2] OF тип;

4.2. Операции над массивами. Обращение к элементу массива в Паскале и алгоритмическом языке записывается одинаково: имя [индекс] Индексом может быть любое выражение соответствующего типа.

В Паскале есть операция присваивания массивов одного типа. Например, описание

VAR A, B : ТАБЛИЦА;

позволяет писать

A := B;

Это означает, что все элементы А принимают значения соответствующих элементов В. Допускается также сравнение однотипных массивов на равенство и неравенство.

4.3. Цикл для (FOR). Его мы рекомендуем рассматривать при изучении массивов отдельно от остальных алгоритмических конструкций.

Вспомним цикл для в алгоритмическом языке:

для X от X_нач до X_кон шаг X_шаг

нц

СЕРИЯ

кц

Запись цикла для на Паскале:

FOR X:=X_НАЧ TO X_КОН DO оператор

или

FOR X:=X_НАЧ DOWNTO X_КОН DO оператор

Механизм работы цикла *для* в Паскале и алгоритмическом языке одинаков. Основные особенности паскалевского *для:*

1) шаг задается выбором служебного слова; TO означает шаг 1, DOWNTO — шаг —1, другие значения шага не допускаются;

2) тело цикла включает только один оператор, серия моделируется с помощью составного оператора;

3) управляющая переменная может быть любого перечислимого типа (мы ограничимся использованием типа INTEGER).

В Паскале даны четкие ответы на вопросы, оставшиеся неясными в алгоритмическом языке. Выражения X_НАЧ и X_КОН вычисляются 1 раз, перед началом цикла. Изменять значение управляющей переменной в теле цикла запрещено. После выхода из цикла значение управляющей переменной не определено.

4.4. Литерные массивы (с элементами типа CHAR) рассматриваются как средство представления символьных строк. Это единственный тип массива, для которого допускается присваивание непосредственных значений и сравнение с ними. Например, описание

VAR A : ARRAY [1..7] OF CHAR;

позволяет писать

A:="паскаль"

Но и здесь есть сильное ограничение: длины левой и правой частей должны совпадать, автоматическое выравнивание не предусмотрено. Нет и операций, подобных соединению и вырезке.

4.5. Ввод-вывод массивов. Паскаль не разрешает использовать имя массива как аргумент в операциях ввода-вывода. Массивы можно передавать только поэлементно. Исключение составляют литерные массивы — аналоги литерных величин. Их можно выводить (а в некоторых версиях и вводить) как единое целое.

В качестве примера переведем на Паскаль алгоритм 8 из библиотеки алгоритмов учебного пособия.

```
PROGRAM подсчет;
  CONST N=100;
  VAR
     A : ARRAY [1 .. N] OF INTEGER;
     ОТР, НУЛ, ПОЛ  :  INTEGER;
     I : INTEGER;
BEGIN
  FOR I:=1 TO N
  DO BEGIN
     WRITE ("ВВЕДИТЕ A[", I, "]:");
     READLN (A [I])
  END;
```

```
    ОТР:=0; ПОЛ:=0; НУЛ:=0;
    FOR I: =1 TO N DO
        IF      A[I]<0  THEN  ОТР:=ОТР+1
        ELSE IF A[I]=0  THEN НУЛ:=НУЛ+1
        ELSE                  ПОЛ:=ПОЛ+1
    WRITELN ("В ТАБЛИЦЕ ", ОТР, " ОТРИЦАТЕЛЬНЫХ ЭЛЕМЕНТОВ, ", НУЛ", НУЛЕВЫХ,
            ", ПОЛ", ПОЛОЖИТЕЛЬНЫХ.")
END.
```

5. Вспомогательные алгоритмы. В алгоритмическом языке любой алгоритм мог исполняться самостоятельно, а мог использоваться как вспомогательный. В Паскале программы явно отличаются от процедур, но процедура представляет собой «программу в миниатюре».

Как и программа, процедура состоит из заголовка, описаний и тела. Полный текст процедуры «вкладывается» в вызывающую программу, среди описаний создается раздел процедур.

5.1. Заголовок процедуры имеет вид

PROCEDURE имя (параметры);

Тип и вид параметров указываются в общем списке. Тип задается сразу после имени параметра (отделяется от него двоеточием), перед именем параметра-результата ставится служебное слово VAR. Аргументы специально не выделяются. Параметры одного типа и вида можно объединять в группы. Между группами ставится точка с запятой.

Для примера рассмотрим заголовок алгоритма и соответствующей ему Паскаль-процедуры.

```
алг ПРИМЕР (вещ A, B, C, цел X, Y)
    арг A, B, X
    рез C, Y
```

```
PROCEDURE пример (A, B : REAL;
                  X : INTEGER;
                  VAR C : REAL;
                  VAR Y : INTEGER);
```

Правила вызова процедур и согласования параметров в Паскале и алгоритмическом языке совпадают.

5.2. Описания в процедурах. В процедуру можно включать описания любых объектов: констант, типов, переменных, других процедур. Объекты, описанные внутри процедуры, не могут использоваться вне ее, они называются локальными для данной процедуры.

В процедурах можно использовать объекты, описанные на внешнем уровне (в объемлющей процедуре или главной программе) до текста данной процедуры. Такие объекты называют глобальными. Обратите внимание: именно доступ к глобальным объектам (в данном случае типам) позволяет использовать параметры нестандартных типов.

Понятия глобального и локального неявно присутствовали и в алгоритмическом языке. Там программист мог создавать два вида объектов: алгоритмы и переменные. Все алгоритмы рассматривались как глобальные объекты (любой алгоритм можно вызвать из любого другого без дополнительного описания), все переменные — как локальные (переменная доступна только в том алгоритме, в котором она описана).

В Паскале программист сам определяет, где будет доступен каждый объект. Такая свобода предоставляет много возможностей, пользоваться которыми следует очень осторожно.

Детальные правила описания и взаимодействия глобальных и локальных объектов, принципы их разумного использования — тема для отдельного обсуждения. Здесь же хотелось бы дать несколько советов для начинающих.

Используйте только глобальные константы и типы. Это позволяет рассматривать их как параметры программы в целом, употреблять во всех процедурах без дополнительного описания.

На первых порах используйте только локальные переменные. Взаимодействие процедур должно происходить только через параметры. Это соответствует правилам алгоритмического языка и исключает связанные с глобальными переменными побочные эффекты, затрудняющие понимание программы.

Не вкладывайте процедуры друг в друга. Вам будет проще разобраться в их взаимодействии, если все они будут включены непосредственно в главную программу. Но помните, что если процедура A вызывает процедуру B, то B должна быть описана раньше A.

5.3. Тело процедуры отличается от тела основной программы лишь завершающим знаком препинания. В программе это — точка, в процедуре — точка с запятой.

5.4. Функции в Паскале играют ту же роль, что и в алгоритмическом языке. Заголовок функции имеет вид

FUNCTION имя (параметры) : тип_функции;

Параметры записываются так же, как в процедурах. В некоторых версиях в параметрах функций нельзя использовать слово VAR. Это соответствует правилу, принятому

в алгоритмическом языке: параметры функций могут быть только аргументами.

В теле функции ее имя используется в двух смыслах. В левой части присваивания оно означает результат функции (аналог слова знач), во всех остальных случаях — рекурсивный вызов функции. Такое правило провоцирует ошибки. Чтобы избежать их, можно ввести отдельную переменную (удобно дать ей имя знач) для получения результата функции, а в конце присвоить имени функции значение этой переменной.

Правила использования описаний, глобальных и локальных объектов в процедурах и функциях совпадают.

Рассмотрим пример перевода на Паскаль алгоритма, содержащего вызов.

```
алг СРЕДНЕЕ (цел N, вещ таб A[1:N] вещ S)
    арг N, A
    рез S
нач
    СУММА (N, A, S)
    S:=S/N
кон
алг СУММА (цел K, вещ таб X [1:K], вещ Y)
    арг K, X
    рез Y
нач цел I
    Y:=0
    для I от 1 до K
    нц
      Y:=Y+X[I]
    кц
кон
PROGRAM среднее;
    CONST N=100;
    TYPE ТАБЛИЦА=ARRAY [1..N] OF REAL;
    VAR
      A : ТАБЛИЦА;
      S : REAL;
      I : INTEGER;
    PROCEDURE СУММА (X : ТАБЛИЦА;
    VAP Y : REAL);
    VAR I : INTEGER;
    BEGIN
      Y:=0;
FOR I:=1TO N DO
    Y:=Y+X[I]
    END;
BEGIN (* начало главной программы *)
    FOR I:=1 TO N
    DO BEGIN
WRITE ("ВВЕДИТЕ A [", I, "]:");
READLN (A[I])
    END;
    СУММА (A, S)
    WRITELN ("СРЕДНЕЕ АРИФМЕТИЧЕСКОЕ
              РАВНО", S/N)
END.
```

Обратите внимание, что размер таблицы исключен из списка параметров: он включен в описание типа «таблица».

Пояснения требует использование переменной I. В алгоритме СРЕДНЕЕ ее не было. В программу она введена для организации цикла ввода. В программе и процедуре I — разные переменные, имена которых случайно совпали.

6. Дополнительные возможности Паскаля связаны в первую очередь с развитой структурой данных. Перечислим темы для дополнительных занятий.

Множества и операции над ними.

Перечислимые типы программиста.

Записи.

Работа с файлами, в том числе текстовыми. Функции EOF и EOLN.

Динамические структуры данных.

Я. ГОЛЬЦ, А. ДОДОНОВ
НИИ СиМО АПН СССР

Методические рекомендации

Знакомство с языком программирования

Основная цель раздела — закрепить теоретические знания и привить учащимся практические навыки по записи алгоритмов на одном из языков программирования. Развить представление о языке программирования как о средстве записи алгоритмов обработки информации для передачи их исполнителю — ЭВМ.

Учащиеся должны иметь общее представление о языке программирования, его назначении, сферах применения, уметь правильно записывать на нем простейшие алгоритмы, запускать их на выполнение на конкретный ЭВМ.

При проведении занятий учитель должен иметь в виду, что подавляющему большинству учеников никогда не придется программировать. В основном они будут пользоваться готовым программным обеспечением. Язык программирования в учебнике излагается в качестве примера средства записи алгоритмов для исполнителя — ЭВМ. Учителю следует обратить основное внимание учеников на «информационный» аспект

темы, показать, как на ЭВМ преобразуется с помощью программы числовая, текстовая, графическая информация.

Сейчас школы оснащены преимущественно дисплеями 3-го класса, работать за которыми в режиме набора текста, согласно санитарно-гигиеническим рекомендациям, можно не более 20 мин в неделю. Это накладывает определенные ограничения на методику проведения практических занятий.

Методически более правильным будет предложить учащимся уже написанные, готовые программы ранее разбиравшихся алгоритмов. Эти программы можно «прокручивать» с различными данными, модифицировать, следя за работой конструкций языка. Длина программ должна быть небольшой — 10—15 команд. Для экономии времени и охраны здоровья учеников желательно передавать заранее заготовленные программы по локальной сети с учительской на ученические машины.

Большим подспорьем на уроке является демонстрационный дисплей (телевизор с экраном 61 см по диагонали), подсоединенный к учительской машине. Все изменения в программе оперативно отражаются на демонстрационном экране, возможно коллективное обсуждение результатов.

При выборе языка программирования (Бейсик или Рапира) следует учитывать, что у Рапиры бóльшая преемственность с алгоритмическим языком, совпадение большинства конструкций (ЕСЛИ, ПОКА, ДЛЯ и т. д.), русскоязычная лексика, понятная большинству учащихся.

Предлагаемая программа для изучения Рапиры довольно сильно расходится с примерным планом практических занятий, приведенным в методическом письме «О преподавании курса ОИВТ в объеме 68 часов». Это связано с особенностями структуры Рапиры и с тем, что практическое ее изучение целесообразно построить ближе к схеме изложения УП, так как литература или какие-либо описания к Рапире малодоступны, в отличие от широко распространенного Бейсика. Программа для Бейсика в основных частях совпадает с примерным планом.

Язык программирования Рапира

Основные конструкции языка описаны в УП, часть II, с. 57—70. Подробные методические рекомендации по проведению занятий даны в методическом пособии для учителей и преподавателей средних учебных заведений, часть II, с. 122—148.

Практические занятия по Рапире предлагается проводить по следующей программе.

1-й урок. Общий вид программы на Рапире. Запись констант, переменных и арифметических выражений на Рапире. Команда присваивания. Команда вывода.

2-й урок. Команды ввода на Рапире.

3-й урок. Составление алгоритмов с использованием команд ветвления Рапиры ЕСЛИ—ТО—ИНАЧЕ—ВСЕ, ВЫБОР—ПРИ—ВСЕ.

4-й урок. Составление алгоритмов с использованием команд цикла Рапиры ПОКА—НЦ—КЦ, ДЛЯ—ОТ—ДО—НЦ—КЦ.

5-й урок. Вызов вспомогательных алгоритмов на Рапире. Алгоритмы вычисления значений функций.

6-й урок. Табличные величины (кортежи) в Рапире.

7-й урок. Решение задач на ЭВМ с использованием различных команд Рапиры.

8-й урок. Проверочная работа.

На первом занятии необходимо на практике дать возможность учащимся исполнить на ЭВМ простейшие программы на Рапире. Учащиеся практикуются в записи величин посредством клавиатуры, рассматривается простейшая команда присваивания. Для вывода результатов изучается команда вывода.

Второе занятие посвящено командам ввода. Они позволяют оперативно менять аргументы в программе, что необходимо в дальнейшем для «прокрутки» программ с различными начальными данными.

Третье и четвертое занятия посвящены работе с командами ветвления и цикла Рапиры. Они аналогичны командам алгоритмического языка, поэтому обычно учащимся не требуется много времени на их практическое освоение.

Пятое и шестое занятия можно было бы проводить третьим и четвертым, но для составления содержательных алгоритмов вычисления функций и использования кортежей необходимы команды цикла и ветвления.

Седьмое занятие носит обобщающий характер. На нем могут быть предложены более сложные задачи, повторены элементы, вызвавшие затруднения учащихся.

Проверочная работа показывает, насколько хорошо учащиеся усвоили минимальные навыки программирования на Рапире.

Существует несколько версий Рапиры. Мы будем придерживаться версии, изложенной в УП.

Для учета особенностей имеющейся версии необходимо ознакомиться с документацией на нее.

На знакомство с языком программирования в курсе выделено 6 ч теоретических занятий. Практикум по программированию следует проводить, естественно, после со-

ответствующих теоретических занятий. На нем закрепляется теоретический материал, но основное внимание следует уделить отработке практических навыков и умений по составлению программ на Рапире и исполнению их на ЭВМ.

В связи с тем что в основном конструкции Рапиры близки к конструкциям алгоритмического языка, занятия предлагается строить по следующей схеме:

текстуальное сравнение записей одного и того же алгоритма на алгоритмическом языке и Рапире, выявление общего и различий (основная часть этой работы проводится на теоретическом занятии, на практикуме можно напомнить о результатах проведенного сравнения);

изменение текста программы и ее «прокрутка» в различных вариантах, сравнение результатов;

аналогичная работа с другой программой;

самостоятельное составление программы учащимися;

обсуждение итогов занятия.

Эта схема не является жесткой. Учитель может по своему усмотрению вносить в нее изменения. Однако рекомендуем обратить внимание на последний пункт схемы: опыт показывает его особую эффективность. Лучше пожертвовать одной программой, чем обсуждением. Оно полезно и ученикам, и учителю.

1-й урок. Общий вид программы на Рапире. Запись констант, переменных и арифметических выражений. Команда присваивания. Команда вывода.

Основная цель. Практическое знакомство учащихся с представлением алгоритмов на Рапире, записью констант, переменных и арифметических выражений, работой команд присваивания и вывода.

Требования к знаниям и умениям. Учащиеся должны знать, как выглядит программа на Рапире, уметь записывать величины, исполнять линейные программы на ЭВМ.

Методические указания. Начать занятие можно со следующей программы.

```
ПРОЦ ПРИМЕР_1
(* ПРОВЕРКА ФОРМУЛЫ РАЗНОСТИ
        КВАДРАТОВ ДВУХ ЧИСЕЛ *)
(=>А=>, =>Б=>, В=>, Г=>)
НАЧ
   А:=9
   Б:=4
   В:=А**2—Б**2
   Г:=(А+Б)*(А—Б)
   ВЫВОД: А, Б, В, Г
КОН
```

Пусть учащиеся исполнят ее на ЭВМ. Затем предложите им изменить в тексте программы значения А и Б и исполнить программу вновь. Обратите их внимание на то, что одни и те же переменные (А, Б) могут служить и аргументами, и результатами алгоритма.

Рассмотрим еще один пример.

```
ПРОЦ ПРИМЕР_2
(=>Э=>, =>Ю=>, Я=>)
НАЧ
   Э:=5
   Ю:=8
   Я:=Э+Ю
   ВЫВОД: Э, Ю, Я
   ВЫВОД: «Э, Ю, Я»
   ВЫВОД: «Э+Ю»
   ВЫВОД: Э+Ю
КОН
```

В этой программе укажите на отличие числовых и литерных переменных.

Упражнения к занятию. УП, ч. II, § 10, с. 59, 60, №№ 1, 3—5.

2-й урок. Команды ввода на Рапире.

Основная цель. Практическое знакомство учащихся с работой команд ввода на Рапире.

Требования к знаниям и умениям. Учащиеся должны знать, как записываются команды ввода на Рапире, уметь составить программу с использованием команд ввода и исполнить ее на ЭВМ.

Методические указания. Начать занятие можно с рассмотрения программы ПРИМЕР 1. В ней мы меняли значения переменных А и Б, меняя текст программы. Это неудобно. Используем команду ввода.

```
ПРОЦ ПРИМЕР_3
(* ПРОВЕРКА ФОРМУЛЫ РАЗНОСТИ
        КВАДРАТОВ ДВУХ ЧИСЕЛ *)
(=>А=>, =>Б=>, В=>, Г=>)
НАЧ
   ВЫВОД: «ВВЕДИТЕ 1-Е ЧИСЛО»
   ВВОД ДАННЫХ: А
   ВЫВОД: «ВВЕДИТЕ 2-Е ЧИСЛО»
   ВВОД ДАННЫХ: Б
   В:=А**2—Б**2
   Г:=(А+Б)*(А—Б)
   ВЫВОД: А, Б, В, Г
КОН
```

Команда ВВОД ДАННЫХ приостанавливает программу и высвечивает на экране вопросительный знак. ЭВМ «ждет» от пользователя данных. Введенное число присваивается переменной, которая следует за командой ввода.

Упражнения к занятию. УП, ч. II, § 13, с. 70, №№ 1, 2. Задание 4 из предыдущего урока выполнить, задавая X1, Y1, X2, Y2 командой ввода.

3-й урок. Составление алгоритмов с использованием команд ветвления Рапиры ЕСЛИ — ТО — ИНАЧЕ — ВСЕ, ВЫБОР — ПРИ — ВСЕ.

Основная цель. Закрепить теоретические знания и привить практические навыки записи алгоритмов ветвления на Рапире.

Требования к знаниям и умениям. Учащиеся должны знать, как записываются команды ветвления на Рапире, уметь составить и исполнить на ЭВМ программу с командами ветвления.

Методические указания.

```
ПРОЦ ПРИМЕР_4
(*СРАВНЕНИЕ ДВУХ ЧИСЕЛ*)
(=>ПЕРВОЕ,=>ВТОРОЕ)
НАЧ
   ВЫВОД:«ВВЕДИТЕ ПЕРВОЕ ЧИСЛО»
   ВВОД ДАННЫХ:ПЕРВОЕ
   ВЫВОД:«ВВЕДИТЕ ВТОРОЕ ЧИСЛО»
   ВВОД ДАННЫХ:ВТОРОЕ
   ВЫБОР
      ПРИ ПЕРВОЕ>ВТОРОЕ:ВЫВОД:«ПЕРВОЕ ЧИСЛО БОЛЬШЕ ВТОРОГО»
      ПРИ ВТОРОЕ>ПЕРВОЕ:ВЫВОД:«ВТОРОЕ ЧИСЛО БОЛЬШЕ ПЕРВОГО»
      ПРИ ВТОРОЕ=ПЕРВОЕ:ВЫВОД:«ЧИСЛА РАВНЫ»
   ВСЕ
КОН
```


Предложите учащимся исполнить программу для различных значений переменных ПЕРВОЕ и ВТОРОЕ.

Изменим программу.

```
ПРОЦ ПРИМЕР_5
(* СРАВНЕНИЕ ДЛИН ДВУХ СЛОВ *)
(=>ПЕРВОЕ_СЛОВО,=>ВТОРОЕ_СЛОВО)
НАЧ
   ВЫВОД:«ВВЕДИТЕ ПЕРВОЕ СЛОВО»
   ВВОД:ПЕРВОЕ_СЛОВО
   ВЫВОД:«ВВЕДИТЕ ВТОРОЕ СЛОВО»
   ВВОД:ВТОРОЕ_СЛОВО
ПЕРВОЕ:=ДЛИН(ПЕРВОЕ_СЛОВО)
   ВТОРОЕ:=ДЛИН(ВТОРОЕ_СЛОВО)
   ВЫБОР
      ПРИ ПЕРВОЕ>ВТОРОЕ:ВЫВОД:«ПЕРВОЕ СЛОВО ДЛИННЕЕ ВТОРОГО»
      ПРИ ВТОРОЕ>ПЕРВОЕ:ВЫВОД:«ВТОРОЕ СЛОВО  ДЛИННЕЕ ПЕРВОГО»
      ПРИ ВТОРОЕ=ПЕРВОЕ:ВЫВОД:«СЛОВА ОДИНАКОВОЙ ДЛИНЫ»
   ВСЕ
КОН
```

Эти два примера демонстрируют, как практически один и тот же алгоритм может быть применен для обработки числовой и текстовой информации.

Упражнения к занятию. УП, ч. II, §13, с. 70, №№ 3—5, §6, с. 33, № 12.

4-й урок. Составление алгоритмов с использованием команд цикла Рапиры ПОКА—НЦ—КЦ, ДЛЯ—ОТ—ДО—НЦ—КЦ.

Основная цель. Закрепить теоретические знания и привить практические навыки записи алгоритмов цикла на Рапире.

Требования к знаниям и умениям. Учащиеся должны знать, как записываются команды цикла на Рапире, уметь составить и исполнить на ЭВМ программу с использованием команд цикла.

Методические указания. Программы с использованием команд цикла чаще всего обрабатывают табличные величины. Если к моменту этого занятия учащиеся не изучили § 12 «Кортежи на Рапире», то можно предложить им программы с циклами без табличных величин.

```
ПРОЦ ПРИМЕР_6
(* ВЫЧИСЛЕНИЕ ЗНАЧЕНИЙ ФУНКЦИИ X**2—5 *)
(* НА ИНТЕРВАЛЕ ОТ —3 ДО 2 С ШАГОМ 0.1 *)
```

```
(=>X,=>У)
НАЧ
   ДЛЯ X ОТ —3 ДО 2 ШАГ 0.1
   НЦ
    У:=X**2—5
    ВЫВОД:X,У
   КЦ
КОН
```

Эту программу можно изменить, сделать ее более универсальной.

```
ПРОЦ ПРИМЕР_7
(* ВЫЧИСЛЕНИЕ ЗНАЧЕНИЙ ФУНКЦИИ A*X**2+BX+C *)
(=>A,=>B,=>C,=>X_MIN,=>X_MAX,=>Ш_ВЫЧ,=>X=>,=>У)
НАЧ
   ВЫВОД:«ВВЕДИТЕ КОЭФФИЦИЕНТЫ A,B,C»
   ВВОД ДАННЫХ:A,B,C
   ВЫВОД:«ВВЕДИТЕ ОБЛАСТЬ ОПРЕДЕЛЕНИЯ X_MIN,X_MAX»
   ВВОД ДАННЫХ:X_MIN,X_MAX
   ВЫВОД:«ВВЕДИТЕ ШАГ ВЫЧИСЛЕНИЙ Ш_ВЫЧ»
   ВВОД ДАННЫХ:ШАГ
   ДЛЯ X ОТ X_MIN ДО X_MAX ШАГ Ш_ВЫЧ
НЦ
У:=A*X**2+B*X+C
ВЫВОД:X,У
   КЦ
КОН
```


Упражнения к занятию. УП, ч. II, § 7, с. 41, 43, №№ 2, 9—11.

5-й урок. Вызов вспомогательных алгоритмов на Рапире. Алгоритмы вычисления значений функций.

Основная цель. Закрепить теоретические знания и привить практические навыки записи вспомогательных алгоритмов, алгоритмов вычисления значений функций на Рапире.

Требования к знаниями и умениям. Учащиеся должны знать, как вызывается вспомогательный алгоритм, алгоритм вычисления значения функции, уметь составить и исполнить на ЭВМ программу с использованием вспомогательных алгоритмов и алгоритмов вычисления функций.

Методические указания. На этом занятии необходимо напомнить учащимся важный метод решений задач — метод последовательного уточнения алгоритма.

В УП, ч. II, на с. 61 приведен алгоритм вычисления площади треугольника по формуле Герона. В нем считаются известными длины сторон.

Пусть теперь треугольник задан координатами своих вершин. Нужно составить алгоритм нахождения его площади с помощью формулы Герона.

Запишем алгоритм нахождения расстояния между двумя точками плоскости, заданными координатами X1, Y1 и X2, Y2, в виде алгоритма вычисления функции.

```
ФУНК РАССТОЯНИЕ (X1, Y1, X2, Y2)
(* РАССТОЯНИЕ МЕЖДУ ДВУМЯ
                        ТОЧКАМИ *)
НАЧ
   ЗНАЧ:=SQRT((X1—X2)**2+
                  +(Y1—Y2)**2)
КОН
```

Теперь запишем алгоритм нахождения площади, используя вспомогательный алгоритм вычисления функции.

```
ФУНК ПЛОЩАДЬ ТРЕУГОЛЬНИКА
                  (X1, Y1, X2, Y2, X3, Y3)
НАЧ
   A:=РАССТОЯНИЕ (X1, Y1, X2, Y2)
   B:=РАССТОЯНИЕ (X1, Y1, X3, Y3)
   C:=РАССТОЯНИЕ (X3, Y3, X2, Y2)
   ЗНАЧ:=ПЛОЩАДЬ (A, B, C)
КОН
```

Упражнения к занятию. УП, ч. II, § 11, с. 62, №№ 1—5.

6-й урок. Табличные величины (кортежи) в Рапире.

Основная цель. Закрепить теоретические знания и привить практические навыки записи кортежей на Рапире.

Требования к знаниям и умениям. Учащиеся должны знать, как записываются кортежи на Рапире, уметь составить и исполнить на ЭВМ программу с использованием кортежей.

Методические указания. К занятию можно составить кортеж данных учеников класса из элементов такого вида:

Иванов, Владимир, Викторович, 1971, май, 12

С этим «кортежем кортежей» можно проводить разные исследования.

Упражнения к занятию. УП, ч. II, § 12, с. 66, №№ 1—7.

7-й урок. Решение задач на ЭВМ с использованием различных команд Рапиры.

Основная цель. Закрепить практические навыки составления программ на Рапире и исполнения их на ЭВМ.

Требования к знаниям и умениям. Учащиеся должны знать форму записи основных команд Рапиры, уметь с их помощью записывать программу и исполнять ее на ЭВМ.

Упражнения к занятию. УП, ч. II, § 10, с 60, № 6, § 8, №№ 2, 8, 10.

8-й урок. Проверочная работа.

Основная цель. Проверить знания и практические навыки составления программ на Рапире и исполнения их на ЭВМ.

Требования к знаниям и умениям. Учащиеся должны уметь составить алгоритм, записать программу на Рапире и исполнить ее на ЭВМ.

Язык программирования Бейсик

В УП, ч. II, с. 70—77, излагаются основные сведения о языке Бейсик, методические рекомендации приводятся в пособии для учителей и преподавателей средних учебных заведений, ч. II, с. 149—170.

Целесообразно при проведении практических занятий по Бейсику несколько изменить последовательность изложения и придерживаться следующей схемы.

1. Средства создания, редактирования, запуска и отладки программы. Непосредственный и программный режимы. Задания на вычисления в обоих режимах. Основные средства работы с файлами (2 ч).

1-й урок. Общий вид программы на Бейсике. Константы, переменные и табличные величины.

2-й урок. Выражения на Бейсике. Использование стандартных функций в выражениях. Команда присваивания.

2. Особенности имеющейся реализации выбранного для работы языка программирования. Основные команды. Разбор и демонстрация работы готовых программ (3 ч).

3-й урок. Команды ввода-вывода.

4-й урок. Команда ветвления и команда выбора.

5-й урок. Команда повторения.

3. Реализация несложных алгоритмов (2 ч).

6-й урок. Команда обращения к подпрограмме.

7-й урок. Графические команды.

8-й урок. Самостоятельная работа.

Перед началом практической работы следует изложить учащимся основные сведения о Бейсике применительно к предстоящей теме.

Строго говоря, под общим названием «Бейсик» известно целое семейство родственных языков программирования. Они различаются между собой в деталях (порой довольно существенно), но имеют ряд общих черт: одинаковые основные структуры (ветвления, циклы и т. д.), обязательная нумерация строк и, самое главное, возможность работы в непосредственном (диалоговом) режиме.

В данных рекомендациях за основу взята версия языка Бейсик для системы MSX (компьютер «Ямаха»). По ходу изложения будут отмечаться основные отличия другого распространенного диалекта Бейсика — для компьютера ДВК-2.

1-й урок. Общий вид программы на Бейсике. Константы, переменные и табличные величины.

Основная цель. Освоить начальные сведения о Бейсике — алфавит, элементы данных; получить первоначальные навыки работы с интерпретатором языка.

Требования к знаниям и умениям. Учащиеся должны представлять общий вид программы на Бейсике, знать правила записи констант и переменных, их типы, знать простейшую форму команды присваивания и команду описания массива, уметь правильно ввести с клавиатуры простейший текст на Бейсике, знать и уметь пользоваться основными командами интерпретатора, уметь запустить готовую программу, уметь записать комментарий и понимать цель комментариев к программам, понимать смысл основных сообщений интерпретатора.

Методические указания. Непосредственно перед работой с компьютером следует сообщить учащимся ряд дополнительных сведений о Бейсике, опущенных в УП: о рабочих процедурах, понятии логической строки и работе с интерпретатором.

Физическая строка — строка, отображающаяся на экране дисплея (или в тексте распечатки программы); ее длина зависит от конкретной ЭВМ (или устройства — дисплея и т. д.). Логическая строка — последовательность символов, завершающаяся символом «возврат каретки», который не отображается на экране дисплея. Логическая строка может содержать несколько физических строк, но не более 255 символов.

Интерпретатор Бейсика — программа, переводящая текст программы на Бейсике в непосредственно выполняемые компьюте-

ром машинные команды. Как правило, интерпретатор записан в ПЗУ и готов к работе сразу после включения компьютера. Символы Ок (ожидание команды), появившиеся на экране дисплея, означают готовность компьютера к приему команды.

Для ДВК-2М интерпретатор Бейсика приходится специально вызывать с магнитного диска, так как в ПЗУ его нет; признаком готовности к работе служит слово READY на экране дисплея.

Интерпретатор Бейсика работает в двух режимах: непосредственном и программном. Если команда вводится без предшествующего ей номера, он тут же выполняется интерпретатором. Пример:

```
Ок
PRINT 2*2
  4
Ок
```


Вводимая строка может содержать и несколько команд подряд, отделяемых друг от друга двоеточием (в ОС ДВК — обратной наклонной чертой \); длина ее не должна превышать 255 символов. В этом режиме строки не остаются в памяти машины, но значения переменных величин запоминаются.

Программный режим используется для выполнения больших программ, которые могут быть введены в память с магнитного диска (ленты) или с клавиатуры. Строки программ должны иметь номера, в соответствии с которыми они хранятся в памяти и выполняются.

Опишем команды интерпретатора, используемые в программном режиме.

Команда RUN запускает на выполнение программу, хранящуюся в памяти. Для запуска программы, хранящейся на внешнем устройстве (магнитном диске), следует указать ее имя: RUN ⟨имя программы⟩.

Команда LIST выводит на экран дисплея текст программы, хранящейся в памяти. Если требуется высветить на экране только строки, начиная с номера N1 и кончая номером N2, команда должна иметь вид: LIST N1—N2.

Команда NEW стирает записанную в памяти программу, т. е. все введенные строки программы теряются.

Команда LOAD ⟨имя программы⟩ вводит в память с внешнего устройства (магнитного диска) программу с указанным именем. При этом прежнее содержимое памяти стирается.

В версии ОС ДВК для этой цели служит команда OLD ⟨имя программы⟩.

Команда SAVE ⟨имя программы⟩ записывает содержимое памяти на магнитный диск — под указанным именем. При этом содержимое памяти не стирается.

Команда DELETE N1—N2 удаляет все строки с номерами от N1 до N2.

Для замены какой-либо строки программы достаточно ввести с клавиатуры новую строку с тем же номером — тогда прежняя автоматически будет удалена и заменена новой.

Для вставки новой строки ей необходимо присвоить номер, промежуточный между номерами тех строк, между которыми мы хотим ее поместить (строки обычно нумеруются с интервалом 10).

Следует сказать несколько слов о значении комментариев к программе. Необходимо разъяснить, что комментарии не являются выполняемыми командами, что к работе программы они не имеют никакого отношения и предназначены для человека (помогают чтению и пониманию текста программы). Особенно велика роль комментариев в больших и сложных программах.

Упражнение 1. Записать в непосредственном режиме (используя оператор присваивания или оператор PRINT) следующие числа: а) 6,0; б) 7,4567; в) 0,0000034; г) 0,000000000005; д) 1,0000000001; е) $4\cdot10^{15}$; ж) $8{,}95\cdot10^{-8}$.

Упражнение 2. Найти при помощи ЭВМ (в непосредственном режиме) ошибки в записи следующих констант на Бейсике: а) 7,6; б) 63.4—E2; в) 5,0678545E2; г) 8.04E—1.8; д) 8,4E—15; е) —.400000055E3

Упражнение 3. Вычислить в непосредственном режиме значения приведенных выражений при таких значениях переменных:

а) $7{,}5\cdot10^{-2}+a-4/d$; б) $6+x-\frac{4{,}1}{2+f}$;

в) $(9{,}3+a)\cdot(3{,}2-b)$; г) $\frac{x^2-x+2}{x^4-5x^2+20}$.

$x=78$, $a=2.5$, $b=.56$E-3, $d=-0.002$, $f=9$.

2-й урок. Выражения на языке Бейсик. Использование стандартных функций в выражениях. Команда присваивания.

Основная цель. Научить школьников грамотно писать арифметические и строковые выражения на Бейсике, используя стандартные функции; добиться понимания структуры и работы команды присваивания.

Требования к знаниям и умениям. Учащиеся должны знать правила записи выражений, использования знаков операций и скобок, знать различные типы операций — арифметические, операции отношения и логические, уметь пользоваться стандартными арифметическими и простейшими строковыми функциями, уметь найти синтаксическую ошибку в выражении при появлении соответствующего сообщения.

Методические указания. Перед началом практической работы следует более подробно, чем в учебнике, рассказать о правилах записи выражений на языке Бейсик. Выражением считаются имя переменной, константа или комбинация констант, имен переменных, знаков операций и имен стандартных функций.

Операции записываются по определенным правилам. В зависимости от значений входящих в состав выражения аргументов оно принимает вполне определенное значение, которое может иметь любой тип, т. е. целый, вещественный или строковый.

Операции первого типа — арифметические, включающие в себя (в порядке убывания старшинства): возведение в степень ∧, умножение * и деление /, сложение + и вычитание —.

Операции выполняются в порядке убывания старшинства: операции одинакового старшинства (например, умножение и деление) — слева направо. Для изменения порядка выполнения операций используются скобки (только круглые!).

Арифметические операции применяются только к арифметическим величинам — целым или вещественным (следует заметить, что операция склеивания строковых величин обозначается тем же символом +, что и сложение).

Операции второго типа — операции отношения, они используются для сравнения двух величин. Результатом сравнения является логическая константа — *истина* или *ложь*. В Бейсике логическим константам присвоены значения 1 и 0, т. е. выражение, в котором встречаются операции сравнения, также имеет числовое значение!

Логические операции (их нет в версии ОС ДВК!) применяются для комбинирования сравнений. Результатом их также является *истина* или *ложь*.

Если в выражении встречаются операции различных типов, то вначале выполняются арифметические, затем операции отношения, затем логические.

В выражения могут входить также имена стандартных функций. Так как понятие вспомогательного алгоритма-функции ученикам уже известно, следует обратить их внимание на то, что стандартные функции — это уже готовые, «библиотечные», вспомогательные алгоритмы вычисления функций, к которым надо только уметь правильно обратиться.

Они делятся на два вида — числовые и строковые. Следует заметить, что несоответствие типа аргумента типу функции — очень распространенный вид ошибки, поэтому при использовании в выражениях различных стандартных функций нужно очень тщательно следить за подставляемыми аргументами.

Теперь можно дать точное правило записи команды присваивания: слева от знака равенства — имя переменной (или элемента массива), справа — выражение. Если значение выражения имеет строковый тип, а имя переменной — числовой (или наоборот), то появится сообщение об ошибке. В остальных случаях тип значения выражения преобразуется в соответствии с типом переменной.

Упражнение 1. Записать на Бейсике выражения arcsin x, arccos x, arcctg x и вычислить их значения для каких-либо значений x.

Упражнение 2. Написать программу для вычисления значений приведенных выражений при указанных значениях параметров:

а) $\nu = \sqrt{\frac{8RT}{\pi\mu}}$;

б) $E = \frac{b}{2c}\left(1 - \frac{a}{\sqrt{R^2+a^2}}\right)$;

в) $\varphi = \operatorname{arctg}\frac{\omega L - \frac{1}{\omega C}}{R}$;

г) $T = \frac{2\pi}{\sqrt{\frac{1}{LC} - \left(\frac{R}{2L}\right)^2}}$;

д) $W = mc\left(\frac{1}{\sqrt{1-\beta^2}} - 1\right)$;

е) $\varphi = -\frac{2\pi x}{y} + \arcsin\left(\frac{y}{\pi A}\sqrt{\frac{W}{2m}}\right)$.

а) $R=7{,}3259$; $T=206{,}8$; $\pi=3{,}14$; $\mu=0{,}0034$;
б) $b=2{,}05\cdot10^5$; $c=2\cdot10^6$; $a=0{,}75$; $R=0{,}9$;
в) $\omega=3{,}8\cdot10^2$; $L=5$; $C=2{,}7\cdot10^{-4}$; $R=20$;
г) $L=2{,}3\cdot10^{-3}$; $C=1{,}3\cdot10^{-6}$; $R=2\cdot10^3$;
д) $m=1{,}6\cdot10^{-24}$; $c=3\cdot10^8$; $\beta=0{,}9$;
е) $x=1$; $y=2{,}9$; $A=1{,}08\cdot10^5$; $W=0{,}6\cdot10^4$; $m=2{,}3\cdot10^{-7}$.

3-й урок. Команды ввода-вывода.

Основная цель. Научить пользоваться командами ввода-вывода при реализации несложных алгоритмов на Бейсике.

Требования к знаниям и умениям. Учащиеся должны знать основные команды ввода-вывода информации и уметь пользоваться ими для ввода исходных данных и параметров программы и вывода результатов в приемлемой форме на экран дисплея (или устройство печати).

Методические указания. Здесь учащимся придется столкнуться с существенным отли-

чием от алгоритмического языка, который не включает никаких средств ввода-вывода информации (так как является лишь формой записи алгоритмов). Всякий же язык программирования обязан иметь такие средства, так как он предполагает обмен информацией между компьютером и человеком. Команды ввода-вывода, как правило, специфичны для каждого конкретного языка (конкретного компьютера).

Бейсик имеет две основные команды ввода-вывода (с некоторыми модификациями в разных версиях) — команду ввода с клавиатуры INPUT и команду вывода на экран дисплея PRINT.

Следует обратить внимание учеников на некоторые тонкости, не отраженные в учебнике. В частности, вариант команды INPUT «с подсказкой» (в версии ОС ДВК такого варианта нет). Ее формат INPUT ⟨подсказка⟩; ⟨список переменных⟩, где ⟨подсказка⟩ — любой вспомогательный текст, поясняющий ситуацию, ⟨список переменных⟩ — имена переменных, разделенные запятыми.

Ввод информации происходит так: сначала на экране появляется ⟨подсказка⟩, затем знак вопроса «?» и через пробел — курсор. Затем интерпретатор Бейсика ожидает ввода информации. Ввод продолжается до нажатия клавиши **ВК**. Если вводится строка из одних пробелов или пустая строка (т. е. сразу нажата клавиша **ВК**), то ни одна из переменных списка ввода не меняет своего значения.

Такой вариант команды ввода позволяет наглядно организовать ввод значений переменных. Пример:

```
INPUT «A=»; A
```

На экране появится

```
A=?
```

после чего компьютер ждет ввода значения переменной A.

В процессе ввода информации довольно часто возникают ошибочные ситуации:

тип величины, приведенной в списке, не соответствует характеру вводимой информации (например, переменная имеет числовой тип, а с клавиатуры пытаются ввести нецифровые символы);

число, вводимое с клавиатуры, не укладывается в допустимый диапазон чисел ЭВМ;

количество вводимых значений (которые разделяются запятыми) превосходит число переменных в списке;

клавиша **ВК** нажата раньше, чем было набрано столько значений, сколько приведено в списке.

Учащихся нужно предупредить о возможности ошибок при вводе и посоветовать им быть как можно аккуратнее.

Основной командой вывода в Бейсике является команда PRINT. О ее формате также следует сказать несколько слов.

Список элементов вывода (среди которых могут быть константы, переменные, выражения) может быть пустым. В этом случае PRINT выводит на экран дисплея пустую строку (т. е. курсор просто перемещается на строчку вниз).

Элементы списка могут отделяться друг от друга запятыми или точками с запятой. Если два элемента списка разделены запятой, то они будут выведены каждый в свою так называемую зону печати. Ширина зоны обычно составляет 14 символов. Если выводимое значение не помещается в одной зоне, то оно продолжается в следующей. Несколько запятых подряд, помещенных в списке вывода, вызывают пропуск соответствующего числа зон. Если же два элемента списка разделены точкой с запятой, при их выводе перехода в следующую зону не происходит — второй элемент выводится непосредственно вслед за первым. Умелое использование соответствующих вариантов оператора вывода позволяет получать на экране результаты в наглядной, удобоваримой форме. Например, оператор

```
PRINT, «Результат=»; X
```

(X=5.61) выдаст значение переменной x в следующем виде:

```
Результат=5.61
```

Упражнение 1. Вывести на экран в одной строке значения трех переменных. Сделать то же с пропуском 2-й зоны.

Упражнение 2. Вывести на экран в одной строке значения переменных A, B, C, D, E с указанием имен переменных.

Упражнение 3. Написать программу, выводящую на экран заголовок и заключительную часть делового письма (Глубокоуважаемый .....Искренне Ваш и т. д.). Переменная часть (имя, отчество и т. д.) задается с помощью оператора ввода.

4-й урок. Команда ветвления и команда выбора.

Основная цель. Ознакомить учащихся с конструкцией и правилами записи команд ветвления и выбора на Бейсике.

Требования к знаниям и умениям. Учащиеся должны знать формат команд ветвления и выбора, уметь записать их без синтаксических ошибок; знать разницу между используемыми в этих командах видами управляющих выражений; понимать отличие соответствующих конструкций алгоритмического языка от команд Бейсика; понимать сообщения об ошибочных ситуациях.

Методические указания. Формат команды ветвления на Бейсике очень близок записи такой же команды на алгоритмическом язы-

ке. Но различия все же есть, и не только синтаксические (английские служебные слова и пр.). Прежде всего это касается «глубины вложенности» ветвления. Если в алгоритмическом языке нет ограничений на количество команд в каждой ветви, то в Бейсике такие ограничения есть: вся команда ветвления (включая и новые ветвления внутри) должна поместиться в одной логической строке (255 символов); после слов THEN и ELSE нельзя помещать слишком длинные серии команд.

В версии ОС ДВК реализована еще более усеченная конструкция — без ветви «иначе» (т. е. без слова ELSE).

Из-за этого, если мы хотим записать большие серии команд в какой-либо ветви, приходится использовать команду обращения к подпрограмме — GOSUB.

В версии ОС ДВК есть и другая модификация команды выбора: IF ⟨условие⟩ GOTO ⟨номер строки⟩. В случае истинности условия начинают выполняться строки с указанного номера.

Такие ограничения, хотя и кажутся непринципиальными, могут все же сильно осложнить запись разветвляющихся алгоритмов на Бейсике.

То же относится и к реализации команды выбора. Если в алгоритмическом языке серия команд записывается непосредственно после условия, стоящего за служебным словом при, то в Бейсике можно лишь осуществить переход (в зависимости от значения управляющего выражения) к той или иной строке (ON ⟨выражение⟩ GOTO ⟨номер строки⟩) или к подпрограмме (ON ⟨выражение⟩ GOSUB ⟨номер строки⟩).

Следует отметить, что все эти ограничения значительно повышают требования к продуманности и аккуратности записи программы на Бейсике по сравнению с алгоритмическим языком.

Упражнение 1. Написать, отладить и исполнить программу решения квадратного уравнения (с выдачей нужных сообщений).

Упражнение 2. Написать программу для решения упражнения 12 § 6 УП, ч. II, с. 33.

5-й урок. Команда повторения.

Основная цель. Продемонстрировать школьникам организацию выполнения циклических процессов в программах на Бейсике.

Требования к знаниям и умениям. Учащиеся должны знать формат команды повторения на Бейсике, уметь записать без синтаксических ошибок простейшую циклическую процедуру, понимать сообщения интерпретатора об ошибочных ситуациях.

Методические указания. В подавляющем большинстве диалектов Бейсика реализована лишь команда повторения с параметром.

Формат этой команды совершенно аналогичен записи команды повторения с параметром алгоритмического языка, только ключевые слова используются английские. Поэтому учащиеся, освоившие команду алгоритмического языка, свободно могут работать с командами Бейсика. Необходима лишь аккуратность в записи, особенно в использовании имени параметра цикла. В случае вложенных циклов нужно следить за тем, чтобы параметры имели различные имена и циклы заканчивались в нужном порядке — сначала внутренний, затем внешний.

Упражнение 1. Написать программу поиска минимального элемента в массиве чисел.

Упражнение 2. Написать программу уточнения корня уравнения $f(x)=0$ (УП, ч. I, с. 95) для функции $f(x)=x^3+2x^2-x-2$ на отрезке [0,2] с заданной точностью $eps=10^{-6}$.

6-й урок. Команда обращения к подпрограмме.

Основная цель. Ознакомить учеников с возможностями организации процедур в программах на Бейсике и с построением программ методом последовательных уточнений.

Требования к знаниям и умениям. Учащиеся должны знать суть метода последовательных уточнений алгоритма (программы), представлять возможности Бейсика для реализации этого метода, уметь использовать команду обращения к подпрограмме, понимать сообщения об ошибках.

Методические указания. Следует напомнить ученикам суть метода последовательных уточнений и объяснить, что он широко используется и при написании программ непосредственно на языке программирования. В частности, он позволяет «распараллелить» работу по написанию большой программы: один человек пишет одну процедуру, другой — другую и т. д. Бейсик допускает использование этого метода — в нем есть команда обращения к подпрограмме, хотя по сравнению с другими языками программирования ее возможности ограничены. Формат этой команды:

GOSUB ⟨номер строки⟩

Она отличается от команды вызова вспомогательного алгоритма в алгоритмическом языке: подпрограмма, представляющая собой запись вспомогательного алгоритма, начинается со строки с номером ⟨номер строки⟩ и заканчивается словом RETURN; она может быть записана в любом месте программы. Подпрограмма на Бейсике не имеет никаких формальных аргументов и результатов.

Имя переменной, используемое в основной программе и в подпрограмме, всегда обозначает одну и ту же переменную (в отличие от алгоритмического языка).

Упражнение 1. Написать программу вывода на экран графика какой-либо функции, используя процедуру вычисления значений этой функции в качестве подпрограммы.

Упражнение 2. Написать программу упорядочения по возрастанию массива чисел, используя в качестве подпрограммы программу поиска минимального элемента.

7-й урок. Графические команды.

Основная цель. Ознакомить учащихся с графическими возможностями современных компьютеров.

Требования к знаниям и умениям. Учащиеся должны знать основные графические команды Бейсика, уметь применить их для построения изображения, иметь представление о различных графических режимах работы дисплея.

Методические указания. Содержание этого раздела касается только компьютера «Ямаха», так как знакомство с графическими командами целесообразно только на ЭВМ с достаточно развитой графикой.

Практика показывает, что графические команды довольно легко усваиваются учениками. Желательно увеличить время практической работы на машине — дать возможность просто порисовать, поэтому можно предоставить ученикам больше самостоятельности: вкратце изложив основные графические возможности Бейсика, предложить им нарисовать несложную картинку.

Команда SCREEN задает режим экрана, SCREEN0 и SCREEN1 — текстовые режимы, SCREEN2 и SCREEN3 — графические. В текстовом режиме обычно нельзя получить графическое изображение, в графическом — вывод текста требует специальной процедуры.

Команда COLOR задает цвет изображения. Различные цвета пронумерованы числами от 0 до 15. Номер цвета может задаваться арифметическим выражением.

Команда PSET ставит точку на экране (в графическом режиме). Точка задается двумя координатами. Координаты должны лежать в допустимых пределах: 0—255 для X и 0—101 для У. Начало координат в левом верхнем углу экрана.

Команда LINE рисует отрезки, прямоугольники и закрашенные прямоугольники.

Команда CIRCLE рисует окружность или дугу окружности, углы задаются в радианах. Следует отметить, что окружность, как правило, получается слегка сплюснутой — это зависит от технических характеристик дисплея. То же касается и качества изображения отрезков — сказывается дискретность графики и недостаточная разрешающая способность дисплея. На эти моменты следует обратить внимание учащихся.

Команда PAINT закрашивает замкнутую область, ограниченную некоторым контуром. Координаты, указываемые в команде, должны лежать внутри контура. Цвет закрашивания совпадает с цветом контура. Если нужно закрасить другим цветом, то необходимо указать в команде ⟨цвет контура⟩.

Упражнение 1. Построить на экране: а) треугольник; б) квадрат; в) эллипс.

Упражнение 2. Написать программу, рисующую на экране светофор и «переключающую» цвета в зависимости от введенного с клавиатуры числа.

8-й урок. Самостоятельная работа.

Основная цель. Проверить знания и практические навыки учащихся по написанию программ на Бейсике и отладке их на ЭВМ.

Требования к знаниям и умениям. Учащиеся должны уметь записать алгоритм решения предлагаемой задачи в виде программы на Бейсике и исполнить ее на ЭВМ.

Фото-ЭВМ

ЧТО МОЖЕТ ЭВМ

Не так давно появились фотокамеры, делающие снимки не на обычную фотопленку — длинную ленту, свернутую в рулон, а на пластиковый диск, покрытый фотоэмульсией: кадры на нем располагаются по окружности. Ну, а теперь — новый шаг, да такой новый, что название «фотокамера» придется менять.

В системе VS-101 фирмы «Касио Компьютер», известной своими дешевыми часами, изображение фиксируется тоже на диске, но магнитном. Собственно, VS-101 — специализированный компьютер, преобразующий изображение в электрические сигналы и записывающий их во внешнюю память. На двухдюймовом диске помещается 50 кадров; правда, качество «снимков» пока далеко от совершенства.

Р. ФРЕЙВАЛД,
доктор физ.-мат. наук
Л. РААТЭ, Л. АНДЕРСОНЕ
Латвийская ССР

Задачи и упражнения для I части курса ОИВТ

Предлагаем задачи и упражнения, пригодные как при изложении нового материала, так и для самостоятельной работы. Задачи рассчитаны на средний уровень учащихся.

I. Понятие алгоритма. Формальное исполнение алгоритма

1. Написать алгоритм приготовления «сладкого завтрака», состоящего из следующих продуктов: 3 ст. ложки пшеничной муки, 4 ст. ложки воды, 1 ч. ложка меда, 1/2 лимона, 1 тарелка молока, 1 тарелка ягод. Рецепт приготовления следующий: пшеничную муку замочить на ночь в воде; добавить мед, сок лимона, молока и перемешать, затем добавить ягоды.

Решение:

1) замочить 3 ст. ложки пшеничной муки в 4 ст. ложках воды;
2) замоченную муку оставить на ночь;
3) к замоченной муке добавить 1 ч. ложку меда;
4) добавить сок 1/2 лимона;
5) добавить 1 тарелку молока;
6) все перемешать;
7) добавить одну тарелку ягод.

2. Составить и записать алгоритм известной задачи перевозки крестьянином на противоположный берег реки волка, козы и капусты, учитывая, что с собой в лодку крестьянин может взять либо волка, либо козу, либо капусту и при этом нельзя позволить волку съесть козу, а козе — капусту.

Решение:

1) перевезти козу на противоположный берег;
2) вернуться обратно одному;
3) перевезти на противоположный берег капусту;
4) вернуться обратно с козой;
5) перевезти на противоположный берег волка;
6) вернуться обратно одному;
7) перевезти на противоположный берег козу.

3. Самостоятельно составить алгоритм пути от дома до школы.

4. Самостоятельно составить алгоритм выполнения режима дня.

5. Составить и записать алгоритм вычисления значения y по формуле $y=x^4+4x+5$.

Решение:

1) умножить x на x, обозначить результат A1;
2) умножить A1 на A1, обозначить результат A2;
3) умножить 4 на x, обозначить результат A3;
4) сложить A2 с A3, обозначить результат A4;
5) сложить A4 с 5, считать результат значением y.

6. Составить и записать алгоритм значения функции

$$f(x)=\frac{x^2+3x}{x^2+3x+2}.$$

Исполнить алгоритм при $x=1$.

Решение:

1) $x\cdot x=R1$;
2) $3\cdot x=R2$;
3) $R1+R2=R3$
4) $R3+2=R4$
5) $R3/R4=f(x)$

При $x=1$:
1) $1\cdot 1=1$;
2) $3\cdot 1=3$;
3) $1+3=4$;
4) $4+2=6$;
5) $4/6=2/3$.

7. Сформулировать и записать алгоритм вычисления значения y по формуле

$$y=\frac{5}{x+5}-\frac{x+5}{4x}$$

при условии $x=0$ и $x\neq -5$. Исполнить алгоритм при $x=5$.

Решение:

1) $x+5=A1$;
2) $5/A1=A2$;
3) $4\cdot x=A3$;
4) $A1/A3=A4$;
5) $A2-A4=y$.

При $x=5$:
1) $5+5=10$;
2) $5/10=1/2$;
3) $4\cdot 5=20$;
4) $10/20=1/2$;
5) $1/2-1/2=0$.

8. По заданным алгоритмам написать формулы для вычисления значения y.

А.
1) сложить x с 2, обозначить результат B1;

2) умножить x на x, обозначить результат B2;
3) сложить B2 с 3, обозначить результат B3;
4) разделить B1 на B3, обозначить результат B4;
5) извлечь квадратный корень из B4, обозначить результат B5;
6) разделить 1 на x, обозначить результат B6;
7) сложить B5 с B6, считать результат значением y.

Ответ.

$$y=\sqrt{\frac{x+2}{x^2+3}}+\frac{1}{x}$$

Б.

1) сложить x с 1, обозначить результат A1;
2) сложить A1 с 1, обозначить результат A2;
3) сложить A2 с 1, обозначить результат A3;
4) разделить 1 на A1, обозначить результат A4;
5) разделить 2 на A2, обозначить результат A5;
6) разделить 3 на A3, обозначить результат A6;
7) сложить A4 с A5, обозначить результат A7;
8) сложить A7 с A6, считать результат значением y.

Ответ.

$$y=\frac{1}{x+1}+\frac{2}{x+2}+\frac{3}{x+3}$$

В.

1) умножить x на x, обозначить результат P1;
2) вычесть из P1 4, обозначить результат P2;
3) сложить x с 1, обозначить результат P3;
4) извлечь квадратный корень из P3, обозначить результат P4;
5) умножить P2 на P4, считать результат значением y.

Ответ.

$$y=(x^2-4)\sqrt{x+1}$$

Г.

1) извлечь квадратный корень из a, обозначить результат A1;
2) вычесть из A1 1, обозначить результат A2;
3) вычесть из A1 2, обозначить результат A3;
4) разделить A2 на A3, обозначить результат A4;
5) разделить A3 на A2, обозначить результат A5;
6) сложить A4 с A5, считать результат значением y.

Ответ.

$$y=\frac{\sqrt{a}-1}{\sqrt{a}-2}+\frac{\sqrt{a}-2}{\sqrt{a}-1}$$

Д.

1) разделить 1 на x, обозначить результат K1;
2) умножить x на x, обозначить результат K2;
3) разделить 1 на K2, обозначить результат K3;
4) умножить K2 на x, обозначить результат K4;
5) разделить 1 на K4, обозначить результат K5;
6) умножить K4 на x, обозначить результат K6;
7) разделить 1 на K6, обозначить результат K7;
8) сложить K1 с K3, обозначить результат K8;
9) сложить K8 с K5, обозначить результат K9;
10) сложить K9 с K7, считать результат значением y.

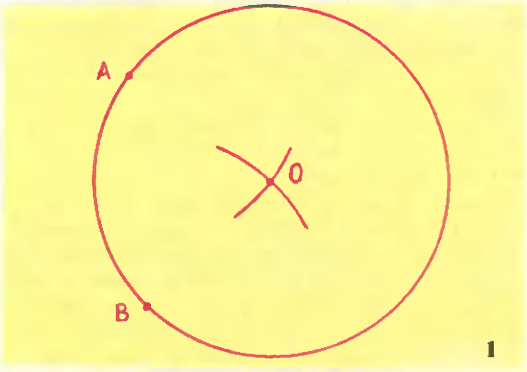
Ответ.

$$y=\frac{1}{x}+\frac{1}{x^2}+\frac{1}{x^3}+\frac{1}{x^4}$$

Перед рассмотрением следующих задач этой темы рекомендуем повторить с учащимися соответствующие геометрические построения.

Под дугой понимаем меньшую из двух дуг, задаваемых двумя точками на окружности.

9. Написать алгоритм построения окружности с данным радиусом r, проходящей через данные точки A и B.



1

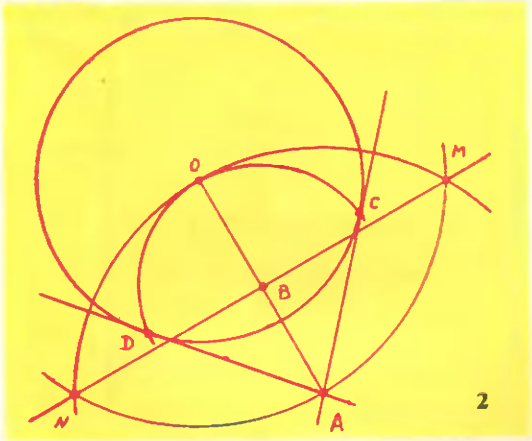
Решение (рис. 1):

1) установить раствор циркуля равным r;
2) поставить ножку циркуля в точку A;
3) провести окружность;
4) поставить ножку циркуля в точку B;
5) провести окружность;
6) одну из точек пересечения окружностей обозначить O;
7) поставить ножку циркуля в точку O;
8) провести окружность.

10. Написать алгоритм проведения касательных к данной окружности через данную точку A, не принадлежащую окружности.

Решение (рис. 2):

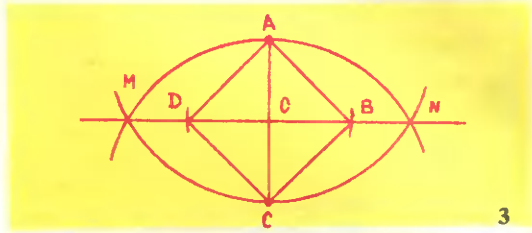
1) провести отрезок OA (O — центр исходной окружности);



2

2) поставить ножку циркуля в точку A;
3) установить раствор циркуля равным длине отрезка OA;
4) провести окружность;
5) поставить ножку циркуля в точку O;
6) провести окружность;
7) точки пересечения построенных окружностей обозначить M и N;
8) провести отрезок MN;
9) точку пересечения отрезков MN и OA обозначить B;
10) поставить ножку циркуля в точку B;
11) установить раствор циркуля равным OB;
12) провести окружность;
13) точки пересечения окружностей с центрами B и O обозначить C и D;
14) провести отрезок CA;
15) провести отрезок AD.

11. Написать алгоритм построения квадрата по данной диагонали CA.



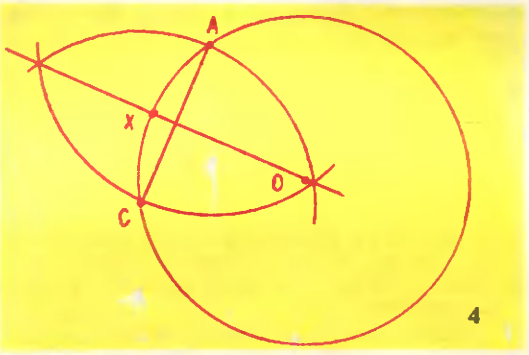
3

Решение (рис. 3):

1) установить ножку циркуля в точку A;
2) установить раствор циркуля равным отрезку AC;
3) провести окружность;
4) поставить ножку циркуля в точку C;
5) провести окружность;
6) точки пересечения окружностей обозначить M и N;
7) провести отрезок MN;
8) точку пересечения AC и MN обозначить O;
9) поставить ножку циркуля в точку O;
10) установить раствор циркуля равным OA;
11) провести окружность;
12) точки пересечения этой окружности с прямой MN обозначить B и D;
13) точки B и D соединить с точками A и C.

12. Выполнить построение по данному алгоритму (даны окружность и точки A и C на ней):
1) провести хорду CA;
2) поставить ножку циркуля в точку A;
3) установить раствор циркуля равным длине отрезка AC;
4) провести окружность;
5) поставить ножку циркуля в точку C;
6) провести окружность;
7) через точки пересечения построенных окружностей провести прямую;
8) точку пересечения прямой с окружностью обозначить X.

Ответ (рис. 4).



4

Вышеприведенные задачи могут быть использованы и при ознакомлении учеников с общими правилами алгоритмического языка.

13. Записать на алгоритмическом языке алгоритм вычисления значения z по формуле

$$z=x^3-3x^2+2.$$

Решение:

```
алг вычисление значения
нач
    умножить x на x, обозначить результат B1
    умножить B1 на x, обозначить результат B2
    умножить 3 на B1, обозначить результат B3
    из B2 вычесть B3, обозначить результат B4
    сложить B4 с 2, считать результат значением z
кон
```

14. В трех коробках A1, A2, A3 находится соответственно 11, 7 и 6 спичек. Составить и записать на алгоритмическом языке алгоритм перемещения спичек из одной коробки в другую с целью выравнивания

количества спичек во всех коробках. Добавлять в коробку можно лишь столько спичек, сколько в ней находится.

Решение:

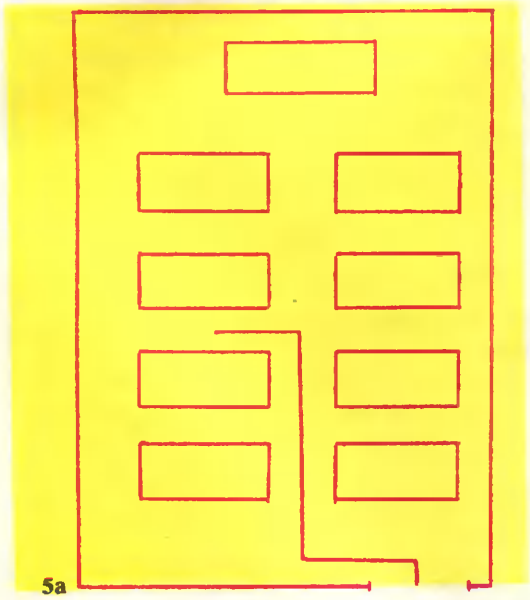
алг перемещение спичек
нач
взять из А1 7 спичек
добавить в А2 7 спичек
взять из А2 6 спичек
добавить в А3 6 спичек
взять из А3 4 спички
добавить в А1 4 спички
кон

При изучении темы «Понятие алгоритма и общие правила алгоритмического языка» рекомендуем под руководством учителя рассмотреть задачи 1, 5, 8а, 8д, 9, 12; для самостоятельной работы учеников в классе использовать задачи 2, 3, 6, 8б, 8в, 10, 13; для работы дома — 4, 7, 8 г, 11, 14.

II. Алгоритмы работы с графической информацией

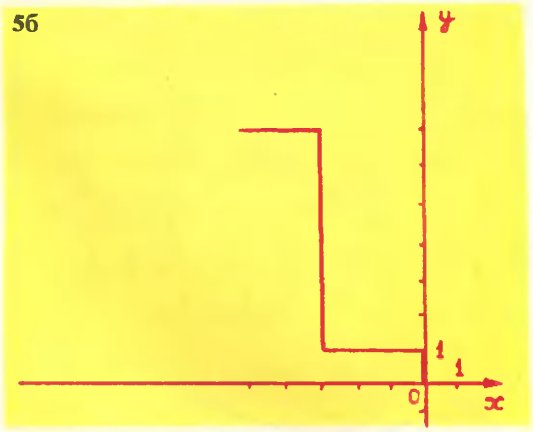
1. Составить, записать и исполнить алгоритм движения от двери до своего места в классе.

Примечание. Разумеется, классы бывают разными. Мы рассмотрим решение для класса, изображенного на рис. 5, а. Дверь находится в точке с координатами (0, 0).

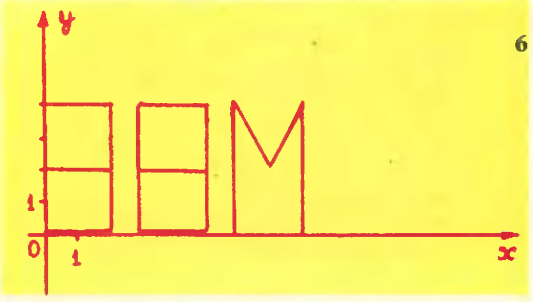


Решение (рис. 5, б):

алг путь
нач
вперед (1)
налево (90)
вперед (3)
направо (90)
вперед (6)
налево (90)
вперед (2)
направо (90)
кон



2. Составить и записать алгоритм рисования слова «ЭВМ» (рис. 6).



Решение:

алг написание слова ЭВМ
нач
не рисуй
вперед (4); рисуй
направо (90); вперед (2)
направо (90); вперед (4)
направо (90); вперед (2)
не рисуй; направо (90)
вперед (2); рисуй
направо (90); вперед (2)
не рисуй; направо (90)
вперед (2); налево (90)
вперед (1); рисуй
налево (90); вперед (4)
направо (90); вперед (2)
направо (90); вперед (4)
направо (90); вперед (2)
не рисуй; направо (90)
вперед (2); рисуй
направо (90); вперед (2)

не рисуй; направо (90)
вперед (2); налево (90)
вперед (1); рисуй
налево (90); вперед (4)
налево (30); назад (2)
направо (60); вперед (2)
налево (30); назад (4)
кон

3. Составить и записать алгоритм рисования индекса своего почтового отделения.

4. Какая фигура будет нарисована при исполнении данного алгоритма?

А.

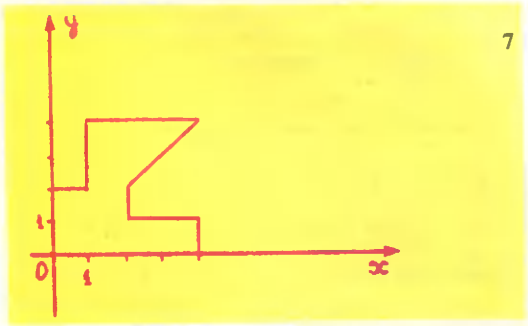
алг рисование фигуры
нач
рисуй; вперед (2)
направо (90); вперед (1)
налево (90); вперед (2)

направо (90); вперед (3)
налево (45); назад ($2\sqrt{2}$)
налево (45); назад (1)
направо (90); вперед (2)
направо (90); вперед (1)
направо (90); вперед (4)
кон

Ответ (рис. 7).

Б.

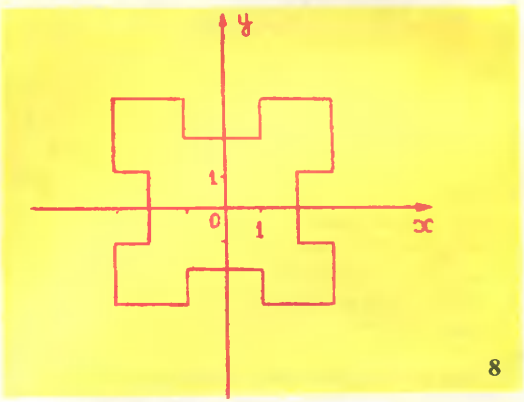
алг рисование фигуры
нач
не рисуй; вперед (2)
рисуй; направо (90)
вперед (1); налево (90)
вперед (1); направо (90)
вперед (2); направо (90)
вперед (2); направо (90)
вперед (1); налево (90)
вперед (2); налево (90)
вперед (1); направо (90)
вперед (2); направо (90)
вперед (2); направо (90)
вперед (1); налево (90)
вперед (2); налево (90)
вперед (1); направо (90)
вперед (2); направо (90)
вперед (2); направо (90)
вперед (1); налево (90)



7

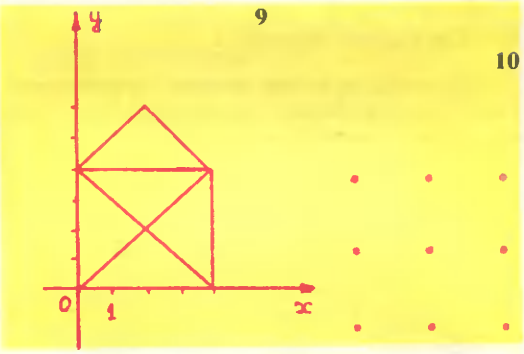
вперед (2); налево (90)
вперед (1); направо (90)
вперед (2); направо (90)
вперед (2); направо (90)
вперед (1); налево (90)
вперед (1)
кон

Ответ (рис. 8).



8

5. Составить и записать такой алгоритм рисования фигуры, изображенной на рис. 9, чтобы ни одна линия не проводилась дважды.



9

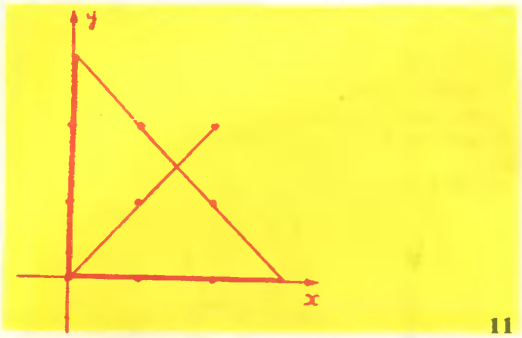
10

Решение:

алг рисование фигуры
нач
рисуй; вперед (4)
направо (90); вперед (4)
налево (45); назад ($4\sqrt{2}$)
направо (45); вперед (4)
налево (45); вперед ($4\sqrt{2}$)
направо (90); вперед ($2\sqrt{2}$)
направо (90); вперед ($2\sqrt{2}$)
направо (45); вперед (4)
кон

6. Составить и записать алгоритм, соединяющий 9 точек (см. рис. 10) четырьмя прямыми линиями, причем каждая проводится только один раз.

Пример решения (рис. 11):



11

алг соединение точек
нач
рисуй; направо (90)
вперед (6); налево (45)
вперед (6√2); направо (45)
назад (6); направо (45)
вперед (4√2)
кон

При изучении темы «Алгоритмы работы с графической информацией» рекомендуем в классе рассмотреть задачи 1, 2, для самостоятельной работы использовать задачи 4а, 4б, для работы дома — 3, 5, 6.

III. Составные команды

1. Составить и записать на алгоритмическом языке алгоритм перехода улицы (движение регулирует светофор).

Решение:

алг переход улицы
нач
если горит зеленый свет
то перейти улицу
иначе стоять
все
кон

2. Прямоугольник, длины сторон a и b которого удовлетворяют равенству

$$a/b=b/(a-b),$$

называют золотым. Записать алгоритм определения золотого прямоугольника.

Решение:

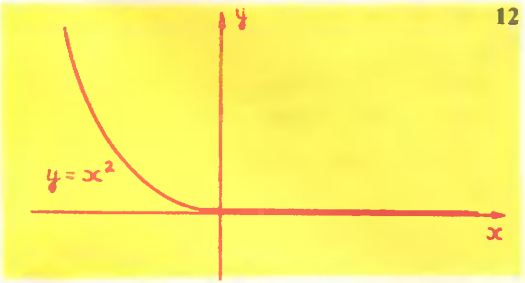
алг определение прямоугольника
нач
разделить a на b, обозначить результат c1
из a вычесть b, обозначить результат c2
разделить b на c2, обозначить результат c3
если c1=c3
то прямоугольник называется золотым
иначе прямоугольник не называется золотым
все
кон

4. Составить и записать алгоритм вычисления значения функции y=f(x) по графику.

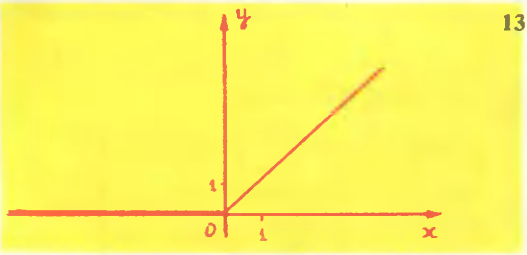
А (рис. 12).

Решение:

алг вычисление значения функции y=f(x)
нач
если x<0
то $y=x^2$
иначе y=0
все
кон



12



13

Б (рис. 13).

Решение:

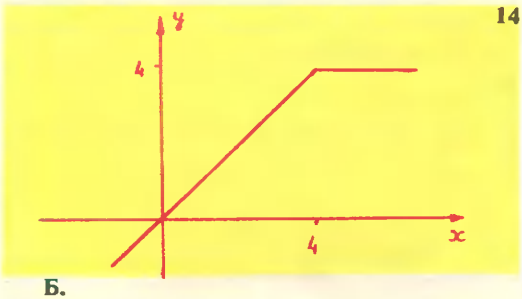
алг вычисление значения функции y=f(x)
нач
если x<0
то y=0
иначе y=x
все
кон

5. Построить график функции y=f(x) по заданному алгоритму.

А.

алг вычисление значения функции y=f(x)
нач
если x<4
то y=x
иначе y=4
все
кон

Ответ (рис.14).

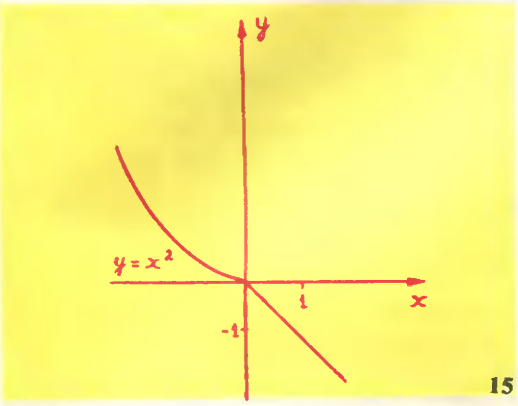


14

Б.

алг вычисление функции
нач
если $x<0$
то $y=x^2$
иначе $y=-x$
все
кон

Ответ (рис. 15).



15

6. Составить алгоритм нахождения большего из трех чисел A, B и C, если $A \neq B \neq C$.

Решение:

алг нахождение наибольшего числа
нач
если A>B
то если A>C
то A наибольшее
иначе C наибольшее
все
иначе если C>B
то C наибольшее
иначе B наибольшее
все
все
кон

Встретились два друга — Янис и Марис. «У меня в одном кармане 3 камешка, а в другом — 2,— сказал Янис.— Сможешь отгадать, сколько камешков в правом, сколько в левом кармане?» Марис предложил Янису умножить число камешков в правом кармане на 3, а в левом — на 2, сложить и назвать полученную сумму. По ней Марис отгадал загадку. Составить алгоритм, с помощью которого Марис решил эту задачу.

Решение:

алг отгадай
нач
если сумма равна 12
то в правом кармане 2 камешка,
в левом — 3 камешка
иначе в правом кармане 3 камешка,
в левом — 2 камешка
все
кон

8. Записать алгоритм, который по характерным признакам химических реакций определяет их тип.

Решение:

алг тип химических реакций
нач
если из одного вещества получается два или несколько веществ
то реакция разложения
иначе если из двух или нескольких веществ образуется одно новое вещество
то реакция соединения
иначе если атомы простого вещества замещают атомы из сложного вещества
то реакция замещения
иначе реакция обмена
все
все
все
кон

9. В корзине 6 яблок. Составить алгоритм раздачи по одному яблоку шести девочкам. Сколько раз исполнится команда повторения в этом алгоритме?

Решение:

алг раздача яблок
нач

```
пока корзинка не пуста
нц
    одно яблоко выдать одной девочке
кц
кон
```

Команда повторения исполнится 6 раз.

10. Составить алгоритм раздачи 6 яблок трем девочкам по два яблока каждой. Сколько раз здесь исполнится команда повторения?

Решение:

```
алг раздача яблок
нач
    пока корзинка не пуста
    нц
        2 яблока выдать одной девочке
    кц
кон
```

Команда повторения исполнится 3 раза.

11. Составить алгоритм раздачи 6 яблок шести девочкам при условии, что каждая получит по одному яблоку и одно яблоко останется в корзинке. Сколько раз исполнится команда повторения?

Решение:

```
алг раздача яблок
нач
    пока в корзинке больше чем одно яблоко
    нц
        одно яблоко выдать одной девочке
    кц
    дать одной девочке яблоко вместе с корзинкой
кон
```

Команда повторения выполнится 5 раз.

12. Составить и записать алгоритм рисования фигуры, изображенной на рисунке.

А (рис. 16).

Решение:

```
алг рисование фигуры
нач
    направо (45)
    пока нарисовано не больше чем 6 отрезков
    нц
        рисуй
        вперед (3√2)
        направо (90)
        вперед (3√2)
        налево (90)
    кц
кон
```

Б (рис. 17).

Решение:

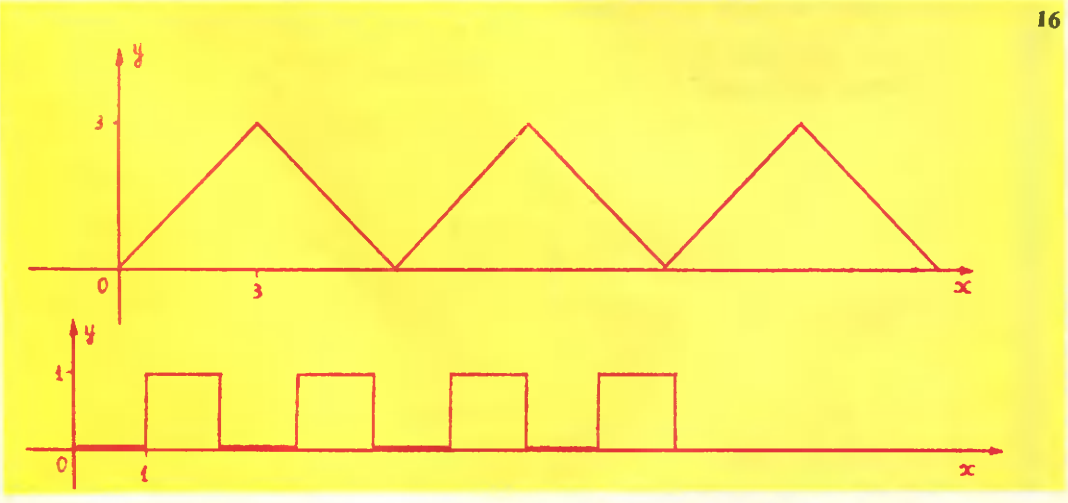
```
алг рисование фигуры
нач
    направо (90)
    пока нарисовано не больше чем 16 отрезков
    нц
        рисуй
        вперед (1); налево (90)
        вперед (1); направо (90);
        вперед (1); направо (90);
        вперед (1); налево (90)
    кц
кон
```

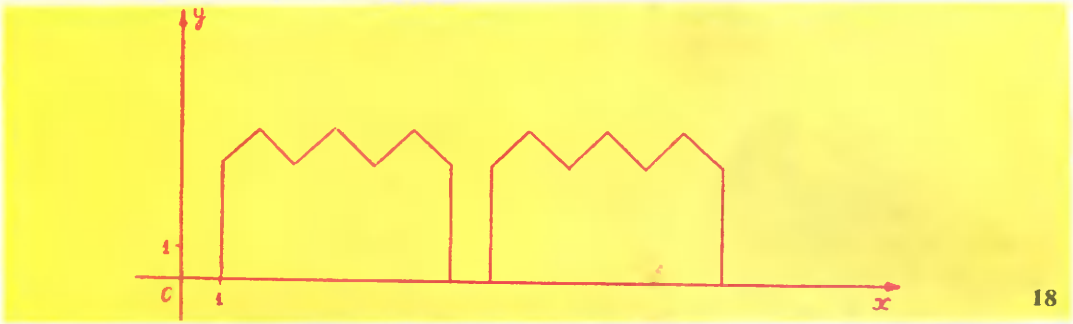
В (рис. 18).

Решение:

```
алг рисование фигуры
нач
    пока нарисовано не больше чем 16 отрезков
    нц
```

16





18

не рисуй; направо (90)
вперед (1); рисуй;
налево (90); вперед (3)
пока нарисовано меньше 6 отрезков
нц
направо (45); вперед ($\sqrt{2}$)
налево (90); назад ($\sqrt{2}$)
направо (45)
кц
назад (3)
кц
кон

13. Какая фигура получится, если исполнить данный алгоритм?

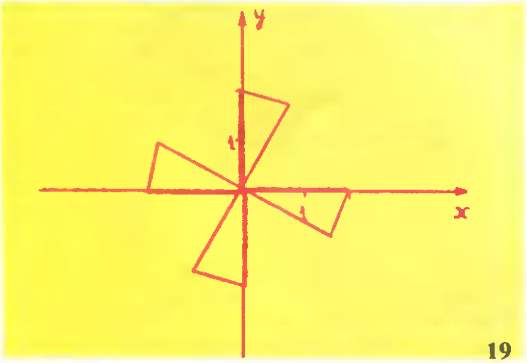
А.

алг рисование фигуры
нач
пока нарисовано не больше чем 12 отрезков
нц
рисуй
направо (30); вперед ($\sqrt{3}$)
налево (90); вперед (1)
направо (60); назад (2)
налево (90)
кц
кон

Ответ (рис. 19).

Б.

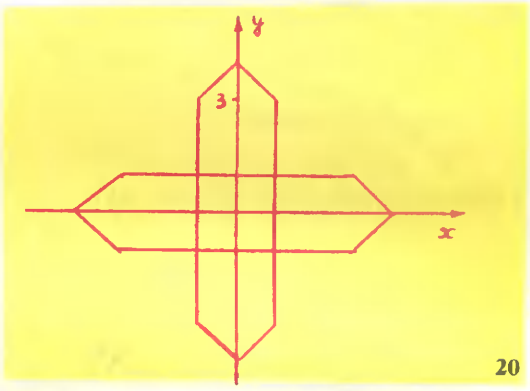
алг рисование фигуры
нач



19

пока нарисовано не больше чем 16 отрезков
нц
не рисуй
направо (90); вперед (1)
рисуй; налево (90)
вперед (3); налево (45)
вперед ($\sqrt{2}$); налево (90)
вперед ($\sqrt{2}$); налево (45)
вперед (3); не рисуй
направо (90); назад (1)
кц
кон

Ответ (рис. 20).



20

14. Группе туристов необходимо переправиться на противоположный берег глубокой и широкой реки. Составить и записать алгоритм их перемещения с помощью двух мальчиков, у которых есть лодка, где помещаются либо мальчики, либо один турист.

Решение:

алг перевоз
нач
пока на первом берегу есть туристы
нц
двум мальчикам плыть на второй берег
первому мальчику остаться на втором берегу
второму мальчику плыть на первый берег
второму мальчику остаться на первом берегу

```
        туристу плыть на второй берег
        первому мальчику плыть на первый берег
    кц
кон
```

15. На стоянке находятся машины красного и синего цветов. Составить и записать алгоритм заезда машин в гараж при условии, что красные машины должны находиться в гараже К, синие — в гараже С.

Решение:

```
алг заезд машин в гараж
нач
    пока стоянка не пустая
    нц
        если машина красного цвета
            то ехать в гараж К
            иначе ехать в гараж С
        все
    кц
кон
```


16. Составить и записать алгоритм отгадывания задуманного целого числа из интервала от 1 до N ($N \geqslant 2$).

Решение:

```
алг отгадывание числа
нач
    пока в интервале больше 2 целых чисел
    нц
        делить интервал пополам
        если задуманное число в первой половине
            то считать первую половину интервалом
            иначе считать вторую половину интервалом
        все
    кц
    если задумано первое число
        то ответ — первое число
        иначе ответ — второе число
    все
кон
```

При изучении темы «Составные команды» для ознакомления с командой ветвления предлагаем в классе рассмотреть задачи 1, 3, 4, 7, для самостоятельной работы и домашних заданий можно использовать задачи 2, 5, 6, 8; для ознакомления с командой повторения в классе рассмотреть задачи 9, 10, 11, 12, 13а, 15, для самостоятельной работы использовать задачи 12, 13б, для домашних заданий — 14, 15, 16.

Т. ПОДДУБНАЯ, И. ФУКС
г. Томск

Построение алгоритмов

Современные методы программирования, основанные на структурном подходе, предусматривают использование специальных приемов. Так, при составлении алгоритма плодотворной оказывается пошаговая детализация: на первом этапе разработки алгоритм рассматривается в общем, как некая совокупность действий для преобразования исходной информации, все последующие этапы — это выявление все более частных его особенностей.

При записи действий на любом этапе нужно учитывать, что исполнителем алгоритма будет ЭВМ, которая умеет совершать только вполне определенные действия, а именно присваивание, ветвление, циклическое повторение. Поэтому любая детализация должна приводить к таким конструкциям.

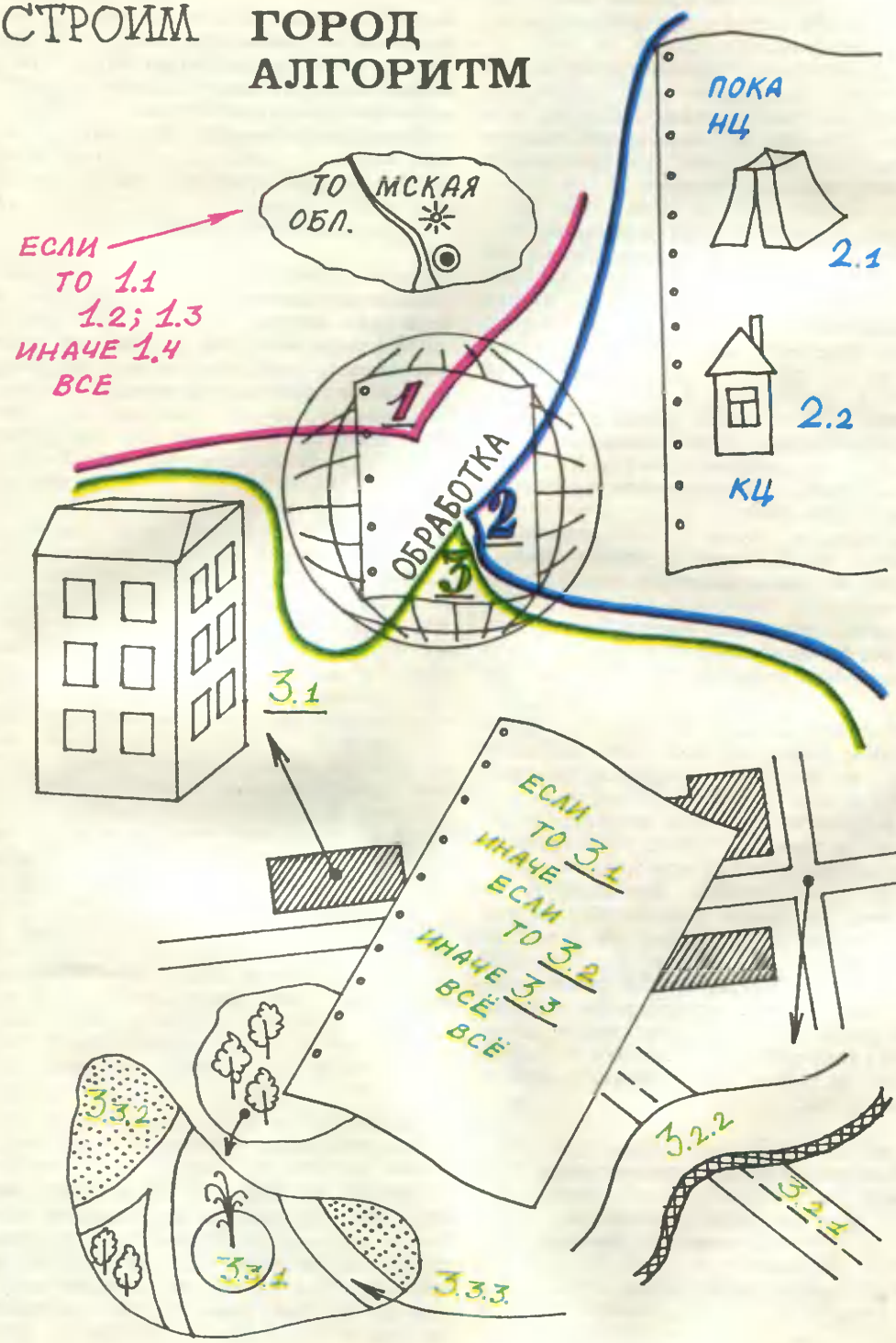
Пояснения к ЛОС 1

«Искусство делать выводы и анализировать, как и все другие искусства, постигается долгим и прилежным трудом, но жизнь слишком коротка, и поэтому ни один смертный не может достичь полного совершенства в этой области. Прежде чем обратиться к моральным и интеллектуальным сторонам дела, которые представляют собою наибольшие трудности, пусть исследователь начнет с решения более простых задач...»

Артур Конан Дойл
«Этюд в багровых тонах»

Можно провести аналогию между разработкой алгоритма решения задачи на ЭВМ и планированием этапов строительства нового города. Обратимся к ЛОС 1. Пусть

СТРОИМ ГОРОД
АЛГОРИТМ



нужно построить новый город. Исходными данными могут быть его желаемые параметры (географические, экономические, социальные и т. д.), обработкой — всё, что связано со строительством, результатом — реально построенный город.

Разделим процесс строительства на три этапа: 1 — изыскания, 2 — подготовка, 3 — стройка. Теперь этапы можно разрабатывать, доводить до частных действий по отдельности. Пользуясь алгоритмической нотацией, запишем, например:

```
1
если место стройки Томская область
то найти площадку для строительства
    на берегу Оби                    1.1
    исследовать грунт                1.2
    разработать проект               1.3
иначе  перенести строительство на
       другой срок                   1.4
все
```


Здесь парами цифр обозначены действия, являющиеся детализацией этапа 1. Каждое из них требует дальнейшего разбора, который снова можно провести независимо от других действий.

Перейдем к этапу 2 — подготовке. Конечно, он состоит из очень многих действий, но мы ограничимся одним:

```
2
пока не все строители имеют жилье
  нц поставить палатку               2.1
     построить времянку              2.2
  кц
```

Это можно описать фразой: «Построить временное жилье для всех строителей», но затем ее все равно пришлось бы представить в виде цикла, поскольку весь процесс разработки алгоритма направлен на получение в конечном итоге последовательности алгоритмических конструкций.

Этап 3 — стройка. Детализируя это действие, мы будем пользоваться результатами этапов 1 и 2, так как они полностью завершатся к его началу. Предположим, что мы строим микрорайон. В какой последовательности возводить отдельные объекты? Если сначала жилые дома, потом оборудовать магистрали, а в последнюю очередь — скверы, то можно детализировать этап 3, например, так:

```
3
если жилья не хватает
  то построить многоквартирный дом   3.1
  иначе
       если движение интенсивное
         то  построить  дорожную
         развязку                    3.2
         иначе разбить сквер         3.3
       все
  все
```

Каждое из новых действий тоже можно детализировать. Например, строительство дома:

```
3.1
выбрать проект                      3.1.1
подготовить стройплощадку           3.1.2
сделать работы нулевого цикла       3.1.3
сложить коробку дома                3.1.4
провести отделочные работы          3.1.5
```

Организация развязки на дороге:

```
3.2
проложить магистраль                3.2.1
построить эстакаду                  3.2.2
```

Озеленение города:

```
3.3
установить фонтан                   3.3.1
разбить цветники                    3.3.2
проложить дорожки                   3.3.3
```

Объединив все шаги детализации в единое целое, получим общую структуру алгоритма строительства города:

```
        ввод
1       если
          то
             1.1
             1.2
             1.3
          иначе
             1.4
        все
2       пока
          нц
             2.1
             2.2
          кц
3       если
          то
             3.1.1
             3.1.2
             3.1.3
             3.1.4
             3.1.5
          иначе если
                  то
                     3.2.1
                     3.2.2
                  иначе
                     3.3.1
                     3.3.2
                     3.3.3
               все
        все
        вывод
```

Пользуясь теми же правилами разработки последовательности действий от общего к частному, следует конструировать алгоритмы решения задач на ЭВМ. Абстрагирование от частностей, мелких подробностей на начальных этапах разработки алгоритма позволяет определить его общее

строение и оформить его таким образом, чтобы в целом следование действий друг за другом образовывало логически законченную структуру.

Использование метода пошаговой детализации позволяет проследить развитие алгоритма вглубь, подчеркнуть, что алгоритм — это последовательность конструкций, каждая из которых может быть достаточно богатой своими «внутренними» действиями. Примерами могут служить организация вложенных циклов или цепочек если—то—иначе.

Пояснения к ЛОС[2]

Почти все алгоритмы содержат циклические конструкции (команды повторения). В них четко выделяются две составные части, несущие разную смысловую нагрузку. Первая указывает, сколько раз должно повториться некоторое действие; вторая — само это действие, т. е., что должно повторяться несколько раз. Первая часть носит название заголовка цикла, а вторая — тела цикла.

На ЛОС 2 для объяснения особенностей построения и оформления циклов используется русская народная сказка «Колобок». Трагическое действие сказки циклично по своей сути. Алгоритмически оно сводится к двум последовательно работающим циклам разного характера. В первом Колобок катится и поет песенку, пока не встретит лису. Во втором цикле существование Колобка прекращается после того, как он 2 раза спел песенку.

Все циклические алгоритмы делятся на две группы. Первая характеризуется тем, что в заголовке цикла задано не конкретное число повторений тела цикла, а некоторое условие, определяющее конец работы цикла. Такой заголовок включает служебное слово пока (WHILE), за которым указывается условие. Если оно истинно, тело цикла выполняется. При такой форме записи число исполнений тела цикла заранее неизвестно. Эта форма универсальна, так как условие может быть любым, например A>10, C=9, B=«ИСТИНА», A>10 и B=«ИСТИНА». Цикл типа пока можно использовать всегда.

Но в целом ряде случаев перед началом работы цикла известно, сколько раз должно повториться его тело. Для удобства записи подобных команд предусмотрен цикл второго типа. С ним связывается некоторая величина — параметр цикла. При работе параметр изменяется от некоторого начального значения (нач. зн) до конечного значения (кон. зн), которые задаются в заголовке цикла; так же может быть указано значение приращения параметра (зн. пр) на каждом шаге цикла. Если оно не указано, то считается равным единице. Тело цикла последовательно выполняется для всех значений параметра; если кон.зн<нач. зн, то цикл не выполняется ни разу. В его оформлении используется служебное слово для (FOR); называется он циклом для, или циклом с параметром. Он не является универсальным, поэтому на ЛОС с ним связано слово ИНОГДА (ИНОГДА в алгоритме можно применять и такую запись).

На ЛОС даны формат записи команд повторения типа пока и типа для, а также их семантика, т. е. расшифровка действий ЭВМ при выполнении этих команд. Для записи семантики используются команды функционального типа («вычислить значение условия»), команды присваивания, ветвления и перехода.

Итак, на ЛОС 2 представлена информация о двух видах циклических алгоритмов.

Важным методическим моментом является параллельное представление, что позволяет сконцентрировать внимание как на общих чертах циклических команд, так и на их различиях.

Для лучшего усвоения этого материала целесообразно написать одни и те же алгоритмы, если они позволяют, с циклами обоего типа. Например, алгоритм для вычисления суммы элементов линейной таблицы:

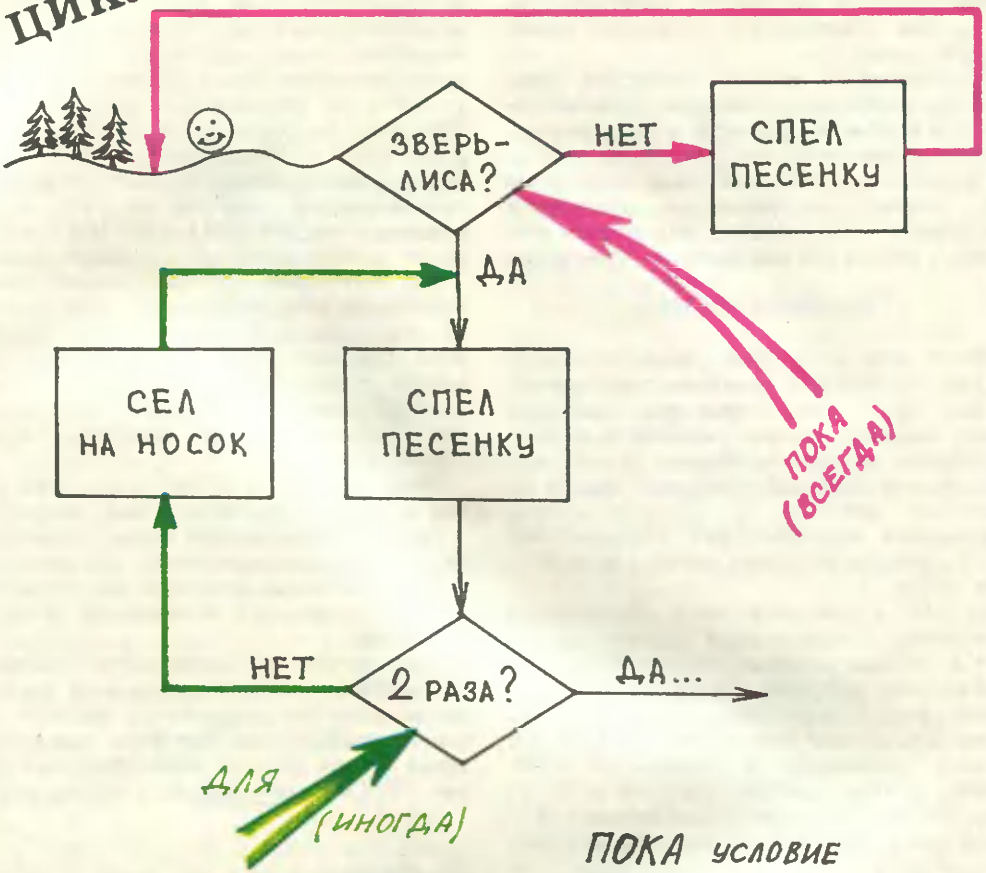
```
алг СУММА (цел таб A [1:1000], цел C)
   арг A;
   рез C
нач цел I
  С циклом пока
I:=1; C:=0
пока I<1000
  нц
    C:=C+A[I]
    I:=I+1
  кц
кон
  С циклом для
C:=0
для I от 1 до 1000 шаг 1
  нц
    C:=C+A[I]
  кц
кон
```

Все величины, которые используются в теле цикла для получения результатов, должны получить свои значения до входа в цикл.

В приведенном выше примере в связи с этим обратите внимание на команды I:=1 и C:=0 в верхней записи и на команду C:=0 в нижней. Почему в нижней этих команд меньше?

ЦИКЛЫ

ЛОС-2



ДЛЯ Р ОТ нач. зн. ДО кон. зн. ШАГ зн. пр.
НЦ
КЦ ТЕЛО ЦИКЛА

ПОКА УСЛОВИЕ
НЦ ТЕЛО ЦИКЛА
КЦ

ЗАГОЛОВОК - СКОЛЬКО?
ТЕЛО - ЧТО?

П.1 Р := НАЧ. ЗН.
П.2 ЕСЛИ Р > КОН. ЗН.
ТО ПЕРЕЙТИ К П3
ИНАЧЕ ВЫПОЛНИТЬ Т.Ц.
Р := Р + ЗН. ПР.
ПЕРЕЙТИ К П2
П.3 СЛЕДУЮЩАЯ КОМАНДА

П.1 ВЫЧИСЛИТЬ УСЛОВИЕ
П.2 ЕСЛИ УСЛ. ИСТИННО
ТО ВЫПОЛНИТЬ Т.Ц.
ПЕРЕЙТИ К П.2
ИНАЧЕ ПЕРЕЙТИ К П.3
П3 СЛЕДУЮЩАЯ КОМАНДА

А теперь рассмотрим несколько примеров, в которых основное внимание уделяется оформлению команд повторения.

Алгоритм вычисления квадратного корня по формуле Герона (метод последовательных приближений)

Алгоритм Герона вычисления квадратного корня из числа A таков:

выбирается некоторое начальное приближение (обычно $B_0=A/2$, но может быть взято любое положительное число);

вычисляется величина $H_0=(A/B_0-B_0)/2$;

вычисляется следующее приближение по формуле $B_{k+1}=B_k+H_k$; $H_k=(A/B_k-B_k)/2$. Процесс вычислений считается законченным, когда H_k становится меньше некоторого числа.

Этот алгоритм достаточно прост, сама формулировка в значительной степени определяет его запись. Для получения результата нужно сделать несколько шагов (итераций). Вычисления на каждой итерации совершаются по одним и тем же формулам, т. е. основной конструкцией в алгоритме будет циклическая. Известно также и условие окончания вычислений, следовательно, нужно использовать цикл пока

Прежде чем переходить к написанию алгоритма, познакомьтесь с интересным методическим приемом, рекомендуемым профессором В. В. Скворцовым (см.: Вестник высшей школы. 1986 г. № 1. С. 34).

«Возможно и обращение к лирике. Вот стихотворение Л. Мартынова (речь в нем идет о девушке):

Кто геометрическое среднее между атомом и Солнцем?
Ты, первое и последнее воплощение красоты.
Не имеющая представления о строении вещества,
слушающая в изумлении эти непонятные слова.
Неспособная принять их к сведению,
будучи ужасно молодой.
Вот ведь, какова ты, нечто среднее
Между атомом и звездой.

Проверим, прав ли поэт. Масса атома водорода $M_a=1{,}67\cdot10^{-27}$ кг, масса Солнца $M_c=1{,}99\cdot10^{30}$ кг. Их произведение равно 3323 кг^2. Поэт утверждает, что масса девушки (в кг) равна квадратному корню из числа 3323. Воспользуемся итеративной формулой Герона, чтобы проверить, прав ли поэт».

Учащимся можно предложить сделать несколько итераций вручную либо с использованием микрокалькулятора. Процесс получения результата (интерес здесь гарантирован) имеет очень большое значение. Учащиеся убеждаются в сходимости процесса независимо от выбора начального приближения; обнаруживают, что приращение H_k может быть отрицательным, и при записи алгоритма уже не забудут воспользоваться функцией, вычисляющей абсолютное значение H_k.

Итак, результат равен 57,645468 кг. А теперь приступим к записи алгоритма.

```
алг корень (вещ A, вещ B)
    арг A
    рез B
нач вещ H; цел I
    B:=A/2; I:=0
    H:=1—A/4
    пока |H|>0.001
      нц B:=B+H
         H:=(A/B—B)/2
         I:=I+1
      кц
кон
```

В него включен подсчет числа итераций (шагов цикла), сделанных для получения результата с заданной точностью. В качестве упражнения полезно видоизменить алгоритм таким образом, чтобы цикл кончал работу:

а) при достижении заданного числа итераций и произвольной точности;

б) при достижении заданного числа итераций или достижении заданной точности.

Задача о «счастливых» билетах

Самый тривиальный алгоритм определения счастливых билетов в заданном диапазоне очевиден:

```
пока не проверили все билеты
  нц
    подсчитать сумму трех левых цифр
    (S)                                  1
    подсчитать сумму трех правых цифр
    (SP)                                 2
    если S=SP
      то напечатать номер счастливого
      билета                             3
    все
  кц
```

Постараемся сократить число проверок номеров билетов.

Предположим, мы подсчитали сумму трех левых цифр номера S. А все ли комбинации цифр в правой части номера нужно исследовать на равенство их суммы значению S?

Возможны три случая: $S<10$; $S\geqslant19$; $10\leqslant S\leqslant18$. Рассмотрим их.

1. Если $S<10$ (например, 5), то присутствие некоторых цифр (в примере — от 6 до 9) в любой позиции в правой тройке исключает ее из рассмотрения, так как заведомо

получится сумма больше S (больше 5). Итак, верхняя граница этих цифр для нашего примера равна 5.

2. Если S≥19, то несколько младших цифр тоже могут оказаться недопустимыми в левой тройке. Например, пусть S=23>19. Число 23 может быть получено при сложении трех цифр следующим образом: 9+9+5; 9+8+6; 9+7+7; 9+6+8; 9+5+9; но 9+4+... — уже неприемлемо. Следовательно, в этом случае минимальное значение цифр в правой тройке равно 23—18=5. Для S=19 минимальное значение равно 19—18=1, для S=20 — 2 и т. д.

3. В случае 10≤S≤18 в правой тройке допустимы любые цифры от 0 до 9.

Займемся детализацией предложенного выше алгоритма. Прежде всего отметим, что, поскольку мы будем выбирать подходящие комбинации цифр в правой части для конкретных сумм цифр левой, внешний цикл можно сделать циклом для с параметром, изменяющимся от INT (N/1000) до INT (M/1000), где N и M — начальный и конечный номера исследуемой серии билетов. Пусть I — число, представляющее собой три левых цифры номера исследуемого билета.

Детализация предложения 1:

```
S:=(I—INT(I/10)*10)+
+(INT(I/10)—INT(I/100)*10)+
+INT(I/100)
```

В предложении 2 необходимо подсчитать сумму трех правых цифр. Для начала определим, какие цифры допустимы в правой части исследуемого номера для того, чтобы он мог быть счастливым. Соответствующий алгоритм таков:

```
L:=0; P:=9
если S<10
  то P:=S
  иначе если S≥19
          то L:=S—18
        все
все
```

А теперь достаточно построить три вложенных друг в друга цикла типа для с параметрами K1, K2, K3, изменяющимися от L до P. Искомая сумма будет подсчитываться как сумма этих параметров.

```
для K1 от L до P
  нц для K2 от L до P
    нц для K3 от L до P
      нц если S=K1+K2+K3
        то C:=(((I*10)+K1)*10+
           +K2)*10+K3
          если C≤M и C≥N
            то напечатать C
          все
      кц
    кц
  кц
кон
```

Предложенный способ подсчета суммы трех правых цифр номера билета привел к тому, что предложение 3 оказалось телом вложенных циклов с параметрами K1, K2, K3.

Поиск простых чисел

Известно несколько методов, позволяющих находить простые числа. Все они основаны на определении простого числа — оно должно делиться только на единицу и само на себя. Таким образом, самый очевидный алгоритм нахождения простых чисел в заданном диапазоне от 1 до G таков:

выбрать очередное число K для анализа;
проверить его делимость на все предыдущие простые числа;
если K делится хотя бы на одно из них
  то оно не простое и нужно перейти к анализу следующего числа из заданного диапазона
  иначе внести K в список простых чисел.

В этом алгоритме есть две циклические конструкции (команды повторения). Первая — выбор очередного числа K из заданного диапазона. Пусть K должно изменяться от 1 до G. Но, поскольку известно, что 1, 2, 3, 5 — простые числа, нижнюю границу для K можно сделать равной 7. Вторая — выбор из уже найденных простых чисел очередного делителя (делимость на единицу проверять не нужно).

Теперь можно детализировать предложенный алгоритм.

```
  команды инициализации цикла выбора K                    1
цикл выбора K из заданного диапазона
  нц команды инициализации цикла проверки делимости K на ранее найденные простые числа   2
  цикл проверки делимости K
    нц если K делится на одно из найденных простых чисел
      то зафиксировать, что K — не простое                 3
      иначе перейти к анализу делимости на следующее ранее найденное простое число        4
    все
  кц
Проверить «признак простоты» числа K; если оно простое, занести в список простых; перейти к анализу следующего числа   5
кц
```

Прежде чем переходить к детализации

предложений 1—5, определим условия окончания работы команд повторения. Пусть простые числа хранятся в массиве с именем P, а I — номер элемента массива, в котором записано значение последнего найденного простого числа. Пусть M — переменная, играющая роль индекса массива, а B — переменная, являющаяся «признаком простоты» анализируемого числа K (если B=0, то K — не простое, если B=1, то K — простое). Тогда условия конца работы циклов могут быть записаны так:

первого (внешнего) — K>G;

второго (внутреннего) — B=0 или M>1 (следует прекратить проверку делимости, если исследуемое число не является простым или индекс массива становится больше номера последнего найденного к этому моменту простого числа).

Теперь детализируем пронумерованные предложения.

```
1   P[1]:=1; P[2]:=2; P[3]:=3; P[4]:=5;
    I:=4
2   B:=1; M:=2 (сначала считаем, что
      анализируемое число является
      простым; проверку делимости начинаем со
      второго простого числа)
3   B:=0
4   M:=M+1
5   если B=1
      то I:=I+1; P[I]:=K
    все
```

На алгоритмическом языке:

```
алг простые числа (цел G, цел таб P[1:G])
    арг G
    рез P
нач цел K, I, M, R, B
    P[1]:=1; P[2]:=2; P[3]:=3; P[4]:=5;
      I:=4
    для K от 7 до G
     нц B:=1; M:=2
       пока B=1 и M≤I
         нц R:=K/P[M]
           если K — R*P[M]=0
             то B:=0
             иначе M:=M+1
           все
         кц
         если B=1
           то I:=I+1; P[I]:=K
         все
    кц
кон
```

Анализ этого алгоритма позволяет сократить число повторений обоих циклов.

Заметим прежде всего, что все простые числа начиная с 5 можно представить как $6n\pm1$ (n=1,2,3,...), что позволяет в цикле по K выбирать для анализа не очередное значение K, а только то, которое представимо этой формулой. Изменим и границы для K: нижняя равна единице (вычисляются простые числа 5 и 7 по формуле $6n\pm1$), а верхняя должна быть взята равной целой части от деления величины G—1 на 6.

Числа типа $6n+1$ не являются ни четными, ни кратными 3. Поэтому проверку делимости следует начинать не с P[2]=2, а с P[4], равного 5.

Можно уменьшить и верхнюю границу параметра M во внутреннем цикле, положив ее равной либо I/2, либо $\sqrt{I}$ (подробное объяснение — на стр. 72).

В измененном алгоритме, приводимом ниже, предусмотрен цикл по переменной T для анализа двух чисел при одном значении K ($6*K+1$ и $6*K-1$). Деление предполагается целочисленным. Для определения числа элементов в массиве P можно воспользоваться оценкой, данной Д. Кнутом: число простых чисел, меньших G, немного больше величины G/lnG. Например:

Значение G	Число простых чисел	G/lnG
10^3	168	145
10^6	78498	72382
10^9	50847534	48254942

Верхняя граница массива P обозначена ЧПЧ.

```
алг ПРОСТЫЕ ЧИСЛА (цел G, цел таб
      P[1:ЧПЧ])
    арг G
    рез P
нач цел K, I, M, R, B, A, T
    P[1]:=1; P[2]:=2; P[3]:=3;
    P[4]:=5; I:=3; G:=(G—1)/6
    для K от 1 до G
     нц A:=6*K—1; T:=1;
     пока T≤2
       нц B:=1; M:=4
         пока B=1 и M<INT (I/2)
           нц R:=A/P[M]
             если A—R*P[M]=0
               то B:=0
               иначе M:=M+1
             все
           кц
       если B=1
         то I:=I+1; P[I]:=A
       все
       T:=T+1; A:=A+2
     кц
    кц
кон
```

Разложение на простые множители

В основе некоторых вычислительных методов лежит тот факт, что каждое положительное число N можно выразить однозначно в виде $N=p_1p_2...p_k$, $p_1 \leqslant p_2 \leqslant ... \leqslant p_k$, где каждое p_i — простое число. К настоящему времени разработан целый ряд интересных приемов, позволяющих получать разложение числа на простые множители. Желающие могут познакомиться с ними по книге Д. Кнута «Искусство программирования для ЭВМ» (М.: Мир, 1972).

Мы рассмотрим здесь решение этой задачи самым очевидным способом. Вот как он описывается у Д. Кнута.

«Если число $N>1$, то его можно последовательно делить на последовательные простые числа $p=2,3,5,...$ до тех пор, пока не будет найдено наименьшее p, для которого $N \bmod p=0$ (остаток от деления N на p равен нулю.— *Т.П., И.Ф.*). Тогда p будет наименьшим простым множителем числа N. Ту же самую процедуру можно применить к $N \leftarrow N/p$ (стрелка $\leftarrow$ означает присваивание значения. — *Т.П., И.Ф.*) — пробовать разделить это новое значение числа N на p и на большие простые числа. Если же на некотором этапе мы находим, что $N \bmod p \neq 0$, то можно сделать вывод, что N — простое число».

Поясним подробнее последний момент, предварительно рассмотрев пример, приведенный у Д. Кнута.

«...В качестве примера работы алгоритма рассмотрим разложение на простые множители числа $N=25852$. Сразу же находим, что $N=2 \cdot 12926$; следовательно, $p=2$. Далее, $12926=2 \cdot 6463$, откуда $p=2$. Имеющееся теперь у нас число 6463 не делится ни на 2, ни на 3, ни на 5,..., ни на 19; находим, что $N=23 \cdot 281$, следовательно, $p=23$. Далее, $281=12 \cdot 23+5$; но $12<23$, следовательно, $p=281$. В рассмотренном примере для определения простых множителей числа 25852 нужно было выполнить 12 операций деления;...»

Таким образом, в задаче нахождения простых множителей некоторого числа процесс деления на простые числа должен быть прекращен тогда, когда при делении числа N на p получается частное R — не целое и меньшее p. Следовательно, если $R=p$ (т. е. $N=p^2$, $p=\sqrt{N}$), то p — максимально возможный простой множитель.

С аналогичной задачей мы встречались в алгоритме поиска простых чисел (во внутреннем цикле алгоритма, в котором очередное исследуемое число делится на все найденные до этого простые числа). Ведь если у N есть делитель $p_1>K$, то следовательно, у него есть и делитель $p_2<K$ такой, что $N=p_1p_2$; p_1 будет «автоматически» найден одновременно с p_2.

Здесь тоже не надо проверять делимость очередного числа на все простые числа, которые меньше его, а следует ограничиться числами, не превышающими целой части $\sqrt{N}=K$.

Теперь приступим к записи алгоритма разложения числа на простые множители по Кнуту.

Очевидно, что весь алгоритм представляет собой цикл, в котором происходит деление на очередное простое число столько раз, сколько это возможно; затем происходит переход к поиску следующего простого множителя. Цикл работает, пока не будут найдены все простые множители числа. Пусть они хранятся в массиве P.

```
Положить переменную M равной заданному числу;                    1
Взять в качестве первого простого множителя K=2;                  2
пока не найдены все простые множители
  нц если остаток от деления числа M на K равен 0
       то занести K в массив P                                    3
          присвоить M значение частного от деления M на K         4
       иначе
          если целая часть частного от деления M на K больше K
          то перейти к поиску следующего множителя                5
          иначе занести M (оно простое) в P                       6
                организовать выход из цикла                      7
       все
  все
кц
```

Теперь приступим к детализации алгоритма. Пусть заданное число имеет имя N.

```
1   M:=N
2   K:=2
```

Обозначим остаток от деления M на K через R, а частное — через C. J — номер элемента в массиве P. В начале тела цикла можно поставить операторы для определения R и C;

```
    C:=целая часть(M/K); R:=M—C*K
3   P[J]:=K; J:=J+1
4   M:=C
5   K:=K+1
6   P[J]:=M
```

Выход из цикла в алгоритме должен произойти тогда, когда найден последний простой множитель. Очевидно, что в этом случае переменная M принимает значение, равное единице. Поэтому условие в заго-

ловке цикла может иметь вид M≠1, а предложение с номером 7 — M:=1.

В результате работы алгоритма будет получен массив P, в котором каждый простой множитель будет встречаться столько раз, сколько он присутствует в представлении числа в виде произведения простых множителей.

```
алг РАЗЛОЖЕНИЕ (цел N, цел таб P[1:N])
    арг N
    рез P
нач цел C, R, J, K
    K:=2; M:=N; J:=1
    пока M≠1
      нц
        C:=целая часть(M/K); R:=M—C*K
        если R=0
          то P[J]:=K; J:=J+1; M:=C
          иначе если C>K
                  то K:=K+1
                  иначе P[J]:=M; M:=1
                все
        все
      кц
кон
```

Алгоритм генерирования перестановок

«А вы, друзья, как ни садитесь,
Все в музыканты не годитесь».

И. А. Крылов

Перестановкой конечного множества из n элементов называется некоторое расположение этих элементов в ряд. Так, множество из трех целых чисел 1, 2, 3 допускает шесть различных перестановок:

1 2 3	2 1 3	3 1 2
1 3 2	2 3 1	3 2 1

В общем случае для множества из n элементов можно построить $n!$ различных перестановок.

Поставим задачу: составить алгоритм, который позволит получить (сгенерировать) все возможные перестановки из n элементов.

Способ 1. Суть его проста. Пусть мы уже построили все перестановки из $(n-1)$ элементов. Тогда, помещая в каждую из них n-й элемент на все возможные места, мы получим все возможные перестановки из n элементов. Поясним это на примере множества чисел 1, 2, 3. Элементы 1 и 2 образуют две перестановки:

1 2 2 1

Добавим к каждой число 3, поставив его на все возможные места:

3 1 2	1 3 2	1 2 3
3 2 1	2 3 1	2 1 3

Получили те же самые перестановки, что приведены выше.

К сожалению, этот простой способ неудобен для реализации на ЭВМ. Здесь предполагается сохранение в памяти всех возможных перестановок. Кроме того, количество элементов в этих перестановках будет меняться, так как мы последовательно добавляем к каждой по одному элементу; следовательно, встает вопрос: в каком виде хранить эти перестановки в памяти ЭВМ? Ответить на него непросто.

Способ 2. Получение перестановок в лексикографическом порядке.

Пусть имеется множество элементов $A=\{a_1, a_2, ..., a_n\}$. Обозначим две разные перестановки этих элементов

$A_i=[a_1^i, a_2^i, ..., a_n^i]$ и $A_j=[a_1^j, a_2^j, ..., a_n^j]$

Определение. Две перестановки называются упорядоченными в лексикографическом смысле, $A_i>A_j$, если $a_1^i>a_1^j$ или существует целое k из $[2, n]$ такое, что $a_l^i=a_l^j$ для всех l из $[1, k-1]$ и $a_k^i>a_k^j$.

Пример 1.

Пусть A= {1,2,3,4 };
A_i= [4,3,2,1];
A_j= [4,2,1,3].

Перестановки A_i и A_j упорядочены лексикографически, $A_i>A_j$, так как выполняются условия $a_1^i=a_1^j$ и $a_2^i>a_2^j$.

Пример 2.

Пусть A={а, б, в, г, д};
A_i= [в, г, а, б, д],
A_j= [а, д, г, б, в].

Если мы введем между элементами множества А соотношение а<б<в<г<д, то можно сказать, что $A_i<A_j$, т. е. эти две перестановки упорядочены в лексикографическом смысле, поскольку a_1^i=в, a_1^j=а и выполняется условие $a_1^i<a_1^j$.

Пример 3. Лексикографически упорядочены фамилии владельцев телефонов в телефонном справочнике.

Вернемся к нашей задаче. Если мы научимся генерировать перестановки в лексикографическом порядке, то сможем получить все возможные варианты — начиная с «самой маленькой» и кончая «самой большой».

Пусть исходное множество, для которого нужно получить все перестановки, есть множество натуральных чисел 1, 2, ..., n. Положим, что исходная перестановка

1, 2, ... , n.

Текущую перестановку обозначим

$A=[a_1, a_2, ... a_n]$.

Предлагается следующая последовательность действий (можно доказать, что она всегда приводит к получению лексикографически упорядоченных перестановок).

1. Просматриваем А справа налево в поисках самого правого i, для которого $a_i<a_{i+1}$. Другими словами, просматриваем перестановку с «хвоста», отыскиваем первый элемент, меньший своего соседа справа, и запоминаем его номер i. Если такого элемента не окажется, т. е. i будет равно 0, значит, эта перестановка последняя и следует закончить выполнение алгоритма.

Заметим, что все элементы, стоящие справа от a_i, упорядочены по убыванию.

2. Просматриваем А слева направо, начиная с a_{i+1}, в поисках элементов a_j таких, что $i<j$ и $a_i<a_j$. Номер (обозначим его j) наименьшего среди этих элементов запоминаем. Иначе говоря, начиная с номера $i+1$ выделяем все элементы, большие найденного a_i, и среди них отыскиваем наименьший.

3. Меняем местами элементы a_i и a_j.

4. Переворачиваем «хвост» перестановки от a_{i+1} до a_n, т. е. элемент a_{i+1} меняем местами с a_n, a_{i+2} — с a_{n-1} и т. д.

Проиллюстрируем работу этого алгоритма на примере перестановок множества из $n=8$ элементов.

Пусть А=[2, 6, 5, 8, 7, 4, 3, 1].

1. Просматриваем перестановку с «хвоста»: $3>1$, $4>3$, $7>4$, $8>7$, $5<8$. Следовательно, $i=3$, $a_3=5$.

2. Просматриваем перестановку начиная с четвертого элемента: $8>5$, $7>5$, $4<5$.

Поскольку «хвост» перестановки упорядочен по убыванию, то дальше проверки делать не нужно. Искомый элемент имеет номер $j=5$, $a_5=7$.

3. Меняем местами a_3 и a_5: $A'=[2, 6, 7, 8, 5, 4, 3, 1]$.

4. Переворачиваем «хвост» перестановки: $A''=[2, 6, 7, 1, 3, 4, 5, 8]$.

Убедимся, что $A<A''$, т. е. перестановки упорядочены лексикографически, для чего сравним попарно элементы этих перестановок: $2=2$, $6=6$, $5<8$.

Приступим к пошаговой разработке алгоритма.

Прежде всего нужно решить вопрос о представлении перестановок в памяти ЭВМ, т. е. вопрос об организации данных. Очевидно, удобно хранить перестановку в массиве, тогда каждый элемент исходного множества будет значением элемента массива. Поскольку мы получаем перестановки последовательно, то одного массива будет достаточно: напечатав перестановку, можно приступать к созданию следующей. Исходные значения элементов этого массива (назовем его ПЕР) — 1, 2, ..., n.

Для получения каждой перестановки мы должны выполнять одни и те же действия, следовательно, основной командой алгоритма будет повторение.

Первый уровень детализации.

```
Заполнить массив ПЕР элементами исходной перестановки, т. е. целыми числами 1, 2, ..., n          1
пока не кончились перестановки                                                                    2
нц
    напечатать массив ПЕР                                                                         2.1
    получить новую перестановку                                                                   2.2
кц
```

Второй уровень детализации.

```
1 • для i от 1 до n
    нц ПЕР [i]:=i
    кц
```

Шаг 2 требует небольшого анализа. Описывая алгоритм в общем виде, мы отметили, что просмотр *последней* перестановки с «хвоста» даст $i=0$, в отличие от всех других перестановок, где $i>0$. Следовательно, условие $i>0$ (или $i\neq 0$) можно сделать условием работы цикла **пока**.

```
2   i:=n
    пока i>0
```

Шаг 2.1 можно детализировать только в программе на конкретном языке программирования.

И наконец шаг 2.2, где мы проверим, не получена ли уже последняя перестановка.

```
2.2
найти самый правый индекс i, для которого
ПЕР [i]<ПЕР [i+1]                                                                                 2.2.1
если i>0 (т. е. перестановка не последняя)
  то справа от ПЕР [i] найти ПЕР [j]>ПЕР [i], причем ПЕР [j] — наименьший среди всех, удовлетворяющих этому условию                                                  2.2.2
     поменять местами ПЕР [i] и ПЕР [j]                                                           2.2.3
     перевернуть «хвост» последовательности справа от i                                           2.2.4
все
```

Третий уровень детализации.

Анализируем шаг 2.2.1. Так как массив просматривается с конца, то сначала $j:=n-1$. Сравнение элементов проведем в цикле, который будет выполняться до тех пор, пока не найдется нужное i. Особо обратим внимание на $j=0$, это условие обязательно нужно проверять, иначе в последней перестановке мы рискуем выйти из массива.

```
2.2.1
j:=n—1
пока ПЕР [j]>ПЕР [j+1] и j>0
  нц
    j:=j—1
  кц
```

Поставив сравнение элементов в заголовок цикла, мы сократили тело цикла до од-

ного действия, перехода к соседнему слева элементу.

Шаг 2.2.2: так как «хвост» перестановки упорядочен, то можно организовать просмотр с $j=n$ и переходить к соседним слева элементам. Искомый — тот, для которого выполнится условие ПЕР [i]<ПЕР [j]. Таким образом, получаем снова циклическую конструкцию:

```
2.2.2
j:=n
пока ПЕР [i]>ПЕР [j]
  нц
    j:=j—1
  кц
```

ПЕР [i] и ПЕР [j] поменяем местами через вспомогательную переменную ВСП:

```
2.2.3
ВСП:=ПЕР [i]
ПЕР [i]:=ПЕР [j]
ПЕР [j]:=ВСП
```

Выполняя шаг 2.2.4, будем продвигаться по массиву с концов к середине. Нам понадобятся две переменные для хранения значений индексов П1 и П2.

Нам нужно поменять местами элементы, симметричные относительно середины отрезка массива, от ПЕР [$i+1$] до ПЕР [n]. Условием работы цикла будет несовпадение значений переменных П1 и П2.

```
2.2.4
П1:=i+1
П2:=n
пока П1<П2
  нц ВСП:=ПЕР [П1]
     ПЕР [П1]:=ПЕР [П2]
     ПЕР [П2]:=ВСП
     П1:=П1+1
     П2:=П2—1
  кц
```

Объединяя вместе все шаги детализации, получим алгоритм генерирования перестановок в лексикографическом порядке.

```
алг ЛЕКС (цел n)
    арг n
нач цел таб ПЕР [1:n]
    цел i, j, П1, П2, ВСП
    для i от 1 до n
      нц
ПЕР [i]:=i
  кц
  i:=n
  пока i>0
    нц
      ПЕЧАТЬ (ПЕР)
      j:=n—1
      пока ПЕР [j]>ПЕР [j+1] и j>0
        нц
          j:=j—1
        кц
      i:=j
      если i>0
        то j:=n
          пока ПЕР [i]>ПЕР [j]
            нц
              j:=j—1
            кц
          ВСП:=ПЕР [i]
          ПЕР [i]:=ПЕР [j]
          ПЕР [j]:=ВСП
          П1:=i+1
          П2:=n
          пока П1<П2
            нц
              ВСП:=ПЕР [П1]
              ПЕР [П1]:=ПЕР [П2]
              ПЕР [П2]:=ВСП
              П1:=П1+1
              П2:=П2—1
            кц
    все
  кц
кон
```


Рекомендации и вопросы

Анализируя этот алгоритм, следует обратить внимание на циклические конструкции. Все они записываются с помощью команды пока, поскольку неизвестно заранее, сколько раз должен проработать цикл.

Очевидно, алгоритм остается работоспособным и тогда, когда элементами исходного множества являются любые величины (не только натуральные числа), между которыми установлены соотношения порядка. Какие изменения нужно внести в алгоритм, чтобы он строил все перестановки множества букв латинского алфавита? Может ли он проделать те же преобразования с буквами русского алфавита?

На примере этого алгоритма полезно продемонстрировать необходимость предварительного оценивания количества результатов, если их вывод производится в цикле. Сколько различных перестановок букв латинского алфавита сгенерирует этот алгоритм?

Л. ЗВАВИЧ
Учитель-методист,

Л. САМОВОЛЬНОВА
Ведущий методист Управления информатики и электронно-вычислительной техники Минпроса СССР

Телевидение для десятиклассников

Две очередные передачи посвящены языкам программирования Рапира и Бейсик и должны обеспечить мотивацию в изучении языков программирования, которая практически отсутствует в условиях безмашинного преподавания. Учащиеся могут писать программы для себя, товарища, учителя, но не для ЭВМ. Телевидение, хотя и в малой степени, позволит им пообщаться с компьютером.

В передаче подробно, на примерах, рассмотрена разница между алгоритмическим языком и языками программирования. В какой-то мере ее можно уподобить разнице между устной и письменной формой изложения. Письменная форма более строгая, формальная, у нее более определенные стилистические законы. Точно так же и языки программирования строятся на более жестких основах, чем алгоритмический язык.

На алгоритмическом языке мы можем общаться друг с другом или запомнить правило решения какой-то задачи. Если оно записано не очень точно или четко, наша сообразительность поможет правильно понять написанное и разобраться в нем.

Язык программирования предназначен для общения с ЭВМ, не имеющей человеческой сообразительности. Инструкция (программа) для нее должна быть написана так, чтобы не было ни двусмысленностей, ни недомолвок, чтобы все было единообразно по форме. В связи с этим мы считаем необходимым привести фрагмент из лекции А. П. Ершова.

«Я позволю себе привести еще одно сравнение. Если алгоритмический язык — это способ рисования карандашом и красками, то Бейсик — это способ рисования путем выкладывания мозаики.

Мозаика состоит из стандартных элементов разных цветов. Их очень легко складывать, приставлять один к другому.

Однако если вы попробуете из мозаики сложить сложную картину, то вы столкнетесь с большими трудностями, вам нужно придумать общий план, найти контуры, только после этого начать деталями и красками заполнять весь рисунок.

Вот примерно такая особенность алгоритмического языка. Вложенные составные конструкции алгоритмического языка — это те же линии и контуры, которые позволяют наметить алгоритмы до того, как вы определили все его детали.

Поэтому можно сказать так: на Бейсике легче учиться писать простые программы, но можно столкнуться с трудностями при написании более сложных программ.

Алгоритмический язык требует более длительного освоения, но зато позволяет легче писать сложные программы».

В передаче о Рапире тщательно разобраны различия и сходства между ней и алгоритмическим языком. Сказано о зависимости выбора того или другого языка программирования от исполнителя (т. е. от ЭВМ), от комплекта внешних устройств ЭВМ и т. д.

При просмотре передачи учащиеся могут, разделив тетрадный лист на две части, выписывать общее и различное в языках программирования и алгоритмическом языке.

В передачах будет показан способ записи программ на Рапире и Бейсике. Учащиеся также увидят в качестве примера программы различных словарных игр, реализованные на этих языках.

Обе передачи просты по форме — это лекции без каких-либо отвлечений, поэтому и просмотр их надо провести просто, без особой подготовки. Однако можно предупредить учащихся о том, что передача очень похожа на лекцию в среднем специальном или высшем учебном заведении, предложить им представить себя студентами, участниками семинара академика А. П. Ершова, тщательно записать текст лекции. Темп рассказа дает такую возможность, особенно если вы смотрите телеуроки регулярно и уже привыкли к ним. Для закрепления и повторения можно предложить кому-либо из школьников выступить на следующем занятии по такому плану:

что мы узнали нового;
какой материал нам стал яснее и понятнее;
что нам показалось лишним и скучным;
что мы не поняли.

При подготовке к просмотру телепередачи учитель может использовать публикации журнала «Информатика и образование».

Конечно, при безмашинном преподавании учащиеся получают только общее представление о языках программирования. Было бы утопией считать возможным обучить их Бейсику или Рапире (тем более что и сам учитель, если он не имел длительной практики работы на ЭВМ, не владеет языком программирования, а уж тем более методикой его преподавания). Полезно, однако, довести до учащихся мысль о том, что изучение основ информатики в безмашинном варианте, и в частности просмотр телепередач, приблизит школьников к ЭВМ, сделает общение с ЭВМ в дальнейшем более простым.

Телепередача «Физические принципы работы ЭВМ» имеет общеобразовательный, обзорный характер и проводится в форме лекции. Ее основные темы:

физические основы передачи и изменения информации в ЭВМ;

принципы сохранения информации.

Эту передачу в основном надо смотреть и слушать. Можно вести и записи в тетрадях, разделив предварительно учащихся на группы:

первая составляет по ходу просмотра план передачи;

вторая — словарь употребляемых терминов, отдельно по информатике, отдельно по физике;

третья — список демонстрирующихся компьютеров и отдельных их устройств;

четвертая конспектирует числовые данные об ЭВМ: скорость передачи информации, объемы памяти и т. п.;

пятая делает список литературы, который приводится во время лекции, фиксирует основные выводы, сделанные лектором;

шестая перечисляет затронутые разделы физики;

седьмая составляет список непонятных слов, услышанных впервые.

Таким образом, в каждой группе будет по 4—5 человек. На обсуждении передачи можно дать слово представителю каждой группы и по результатам ответов выставить семь оценок.

Целесообразно перед просмотром передачи попросить школьников прочесть дома соответствующий параграф (§ 5) пробного учебного пособия. Для организации внимания учащихся можно использовать вопросы, имеющиеся в конце этого параграфа.

Авторы статей о телепередачах по курсу «Основы информатики и вычислительной техники» просят читателей поделиться своим опытом использования передач на уроке, методикой их просмотра и обсуждения.

Напишите нам:

1. Считаете ли вы полезными эти передачи:

а) для учащихся?

б) для учителей?

2. Какие из увиденных передач вы считаете наиболее удачными?

3. Какую из форм телеурока (лекция, фрагмент урока и т. п.) вы считаете наиболее плодотворной?

В. ГУРЕВИЧ

Повышение доступности алгоритма УПОРЯДОЧЕНИЕ

Многие учащиеся испытывают трудности в понимании алгоритма упорядочения линейной таблицы, связанные с тем, что в нем производятся действия над таблицей с изменяющимся номером начального элемента, используется алгоритм МИНЭЛЕМЕНТ, сам по себе сложный, а также серия команд, обеспечивающая обмен местами двух элементов, которую целесообразно выделить в самостоятельный алгоритм. Эта структура порождается способом упорядочения: найти номер наименьшего элемента таблицы C[n : M] и поменять местами этот элемент с начальным, затем начиная со второго элемента получившейся таблицы найти номер наименьшего и поменять местами этот элемент со вторым и т. д.

Такой подход требует предварительного построения алгоритма поиска номера l наименьшего элемента таблицы A[K : N] — МИНЭЛЕМЕНТ (K, N, A, l) и использования его в цикле, параметром которого является номер начального элемента таблицы. Так как номер начального элемента таблицы A[K : N] отличен от 1, у учащихся возникают сложности при определении начальных значений промежуточных переменных;

кроме того, им бывает трудно понять суть действий, выполненных в алгоритме МИНЭЛЕМЕНТ согласно команде

если МИН>A[i]
то МИН: =A [i]; l:=i
все

Материал значительно упростится, если идею упорядочения применять по-другому: найти номер наибольшего элемента таблицы и поменять его местами с последним, затем найти номер наибольшего из всех элементов, исключая последний, и поменять местами найденный элемент с предпоследним и т. д. При таком подходе потребуется предварительно построить алгоритм поиска номера наибольшего элемента таблицы МАКСЭЛЕМЕНТ. Его (в отличие от алгоритма МИНЭЛЕМЕНТ) можно легко построить, используя уже имеющиеся алгоритмы нахождения наибольшего элемента Y линейной таблицы A[1 : n] — НЭЛТ(n, A, y) и нахождения номера K элемента линейной таблицы A[1 : n], равного данному числу l,— ПОИСК (n, A, l, K). Легко видеть, что МАКСЭЛЕМЕНТ состоит из двух команд вызова этих алгоритмов. В самом деле, выполнение команды НЭЛТ (n, A, y) даст наибольший элемент y таблицы A[1, n], последующее выполнение команды ПОИСК (n, A, y, K) даст номер K наибольшего элемента рассматриваемой таблицы.

алг МАКСЭЛЕМЕНТ (нат n, вещ таб A[1 : n], нат K)
арг n, A; рез K
нач вещ y
НЭЛТ (n, A, y); ПОИСК (n, A, y, K)
кон

Предлагаемый подход позволяет ограничиться рассмотрением таблиц, номера начальных элементов которых равны 1, что, не меняя существа дела, упрощает алгоритмы работы с ними. Таким образом, с целью повышения доступности рассматриваемого материала целесообразно вместо алгоритмов МИНЭЛЕМЕНТ и УПОРЯДОЧЕНИЕ (n, M, C), приведенных в пробном учебном пособии, использовать алгоритмы МАКСЭЛЕМЕНТ, ОБМЕН (см.: Авербух А. Упорядочение линейной таблицы // Поурочные разработки по курсу «Основы информатики и вычислительной техники». Ч. I. М., 1986) и УПОРЯДОЧЕНИЕ (n, A):

алг УПОРЯДОЧЕНИЕ (нат n, вещ таб A[1, n])
арг n, A; рез A
нач цел i; i := 0; нат K
пока i<n—1
нц
МАКСЭЛЕМЕНТ (n—i, A, K)
ОБМЕН (A[n—i], A[K])
i := i + 1
кц
кон

Как видим, совершенствование содержания за счет наиболее эффективной реализации внутрипредметных связей — один из путей повышения доступности изучаемого материала.

Другим путем является совершенствование методики изложения учебного материала. В основу предлагаемой ниже методики положен поиск такого плана решения задачи, реализация каждого пункта которого может быть описана на алгоритмическом языке.

Рассказав о практической значимости упорядочения больших наборов чисел, сформулируем задачу: расположить элементы таблицы A[1 : n] по возрастанию.

Поиск решения начнем с того, что предложим учащимся упорядочить таблицу с небольшим числом элементов и рассказать, как они это сделали. Ученики без затруднений таблицу

i	1	2	3	4	5
A[i]	—2	7	4	0	—5

преобразовывают в

i	1	2	3	4	5
A[i]	—5	—2	0	4	7

Выполняемые действия они описывают примерно так: «Взяли наибольшее число 7 и поставили его на последнее место, из оставшихся чисел взяли наибольшее — 4 и поставили его на предпоследнее место и т. д.». Понимания того, что эти действия, вполне понятные человеку, не обладают необходимой точностью, чтобы быть командами алгоритма, еще нет. Надо задать наводящий вопрос: «Когда вы поместили 7 на последнее место, где оказался элемент, находившийся на последнем месте?»

Это помогает ученикам уточнить ответ: для упорядочения таблицы надо поменять местами ее наибольший элемент и последний, затем наибольший элемент из всех, кроме последнего, поменять местами с предпоследним и т. д. Таким образом, для построения алгоритма упорядочения линейной таблицы надо знать, как поменять местами два ее элемента. Теперь подведем учащихся к пониманию того, что для обмена местами элементов таблицы надо знать их номера. Для этого зададим такой вопрос: наибольший элемент таблицы равен 7; с помощью каких команд возможен обмен местами этого элемента с последним? И тут выясняется, что неизвестно, на какое место следует поставить последний элемент таблицы, так как не указан номер наибольшего элемента; приходим к выводу, что надо еще уметь находить номер наибольшего элемента.

В итоге возникает такой план решения задачи упорядочения линейной таблицы:

1) решить подзадачу 1: построить алгоритм определения номера наибольшего элемента линейной таблицы;

2) решить подзадачу 2: построить алгоритм обмена местами двух элементов линейной таблицы;

3) решить основную задачу, используя результаты решения подзадач 1 и 2.

Методику обучения решению подзадач 1 и 2 излагать не будем, так как алгоритм МАКСЭЛЕМЕНТ достаточно прост и при его обсуждении в общих чертах просматривалась и методика обучения, а план построения алгоритма ОБМЕН очевиден.

Рассмотрим методику обучения решению задачи упорядочения линейной таблицы.

Поиск решения начнем с рассмотрения уже построенного плана. Согласно третьему пункту, для упорядочения таблицы надо применить построенные алгоритмы. Как? Посредством команд вызова: МАКСЭЛЕМЕНТ (n, А, К); ОБМЕН (А[К], А[n]).

Что произойдет в результате выполнения этих команд? Будет получена таблица А[1 : n], последний элемент которой А[n] является наибольшим.

Окажется ли таблица упорядоченной? Нет, так как предшествующие последнему n—1 элементов остаются расположены, вообще говоря, произвольным образом.

Как их упорядочить? Очевидно, применив те же алгоритмы для первых n—1 элементов таблицы А. Команды вызова будут теперь выглядеть так: МАКСЭЛЕМЕНТ (n—1, А, К); ОБМЕН (А[К], А[n—1]).

Повторив, если потребуется, эти рассуждения для первых n—2 элементов полученной таблицы, сформулируем обобщенный вывод.

В алгоритме упорядочения линейной таблицы:

1) аргументами являются нат n и вещ таб А[1 : n], результатом — таблица А, состоящая из тех же элементов, но расположенных по возрастанию;

2) содержится команда повторения серии из двух команд вызова вспомогательных алгоритмов МАКСЭЛЕМЕНТ (n—i, А, К) и ОБМЕН (А[К], А[n—i]) и команды наращивания параметра $i := i + 1$;

3) вводятся промежуточные переменные цел i, начальное значение которой равно 0, и нат К;

4) команда повторения выполняется до тех пор, пока число n—i рассматриваемых элементов таблицы А остается больше 1, т. е. пока $n-i>1$, $i<n-1$.

Реализация каждого из этих пунктов приводит к сделанной выше записи алгоритма УПОРЯДОЧЕНИЕ. Отметим, что последовательность реализации этих пунктов при записи алгоритма (1, 3, 4, 2) не совпадает с последовательностью их получения при поиске решения.

А. ЗАК

Развитие самостоятельного мышления школьников

Самостоятельное мышление учащихся в самом общем плане характеризуется умением разобраться в усваиваемом учебном материале без посторонней помощи. В психологическом аспекте успешное самостоятельное выполнение учеником каких-либо заданий обусловлено наличием у него определенных качеств мышления, способствующих умению проанализировать задания, выделяя в первую очередь существенные и необходимые свойства изучаемых объектов (*глубина* мышления).

Необходимым качеством самостоятельного мышления является и его *критичность*, проявляющаяся в степени осознанности процесса мышления и предполагающая умение обосновать правомерность способов решения, используемых при выполнении заданий.

Еще одним качеством, обеспечивающим самостоятельное выполнение действий и заданий, выступает *гибкость* мышления. Обладая гибким умом, человек способен быстро отвлечься от тех связей и отношений в содержании выполняемого задания, которые были установлены им раньше, и легко образовать новые, перейдя от прямых связей к обратным, от одной системы действий к другой.

Развитие глубины, критичности и гибкости мышления учащихся в значительной степени зависит от того, какие методы преподавания реализуются в процессе обучения.

Особую, на наш взгляд, роль в развитии этих качеств у старшеклассников может сыграть обучение их основам информатики и вычислительной техники по соответствующему пособию (Основы информатики и вычислительной техники: Пробное учебное пособие для средних учебных заведений / Под ред. А. П. Ершова, В. М. Монахова. М., 1985. Ч. 1). В пособии имеются задания по разработке идей и проектов алгоритмов для решения целого класса задач. Выполнение этих заданий создает благоприятные возможности для формирования у школьников обобщенности, глубины мышления. Кроме того, курс информатики содержит задания, связанные с установлением пригодности известных алгоритмов к решению данной задачи, с выбором вспомогательных алгоритмов при разработке новых предписаний. Такие задания должны способствовать формированию у школьников осознанности, критичности мышления, поскольку их успешное выполнение требует отчетливого понимания учащимися границ применимости используемых алгоритмов. Формированию гибкости мышления должны помочь задания, требующие преобразования известных алгоритмов (их упрощения, расчленения, изменения команд) соответственно специфике данной задачи и конкретным особенностям исполнителя алгоритма.

При изучении основных понятий курса информатики на первом этапе уместно использовать специальные вопросы, ответы на которые требуют проявления глубины, критичности и гибкости мышления. Рассмотрим такие вопросы на примере усвоения фундаментального понятия курса — понятия «алгоритм».

Целесообразно ставить и обсуждать вопросы, способствующие включению в определение понятия «алгоритм» существенных и специфических свойств предписаний этого типа. Так, вслед за вопросом: «Что такое алгоритм?» — следует задавать вопросы, уточняющие это понятие, например: «Любое ли предписание можно считать алгоритмом?»; «Какими необходимыми свойствами должно обладать предписание, чтобы быть алгоритмом?»; «По каким признакам алгоритм отличается от неалгоритма?»

Следующий этап работы с понятием «алгоритм» связан с организацией деятельности учащихся по практическому уяснению границ его применения.

На основе содержания параграфа «Алгоритм и его свойства» в «Основах информатики и вычислительной техники» мы предлагаем следующее определение: *алгоритм — это предписание,*

содержащее законченный перечень последовательных действий, каждое из которых точно описывает отдельный завершенный шаг решения задачи и понятно исполнителю этого предписания. Можно разработать ряд вариантов одного предписания или ряд разных предписаний (т. е. относящихся к разным задачам), в каждом из которых отсутствует какое-нибудь существенное свойство алгоритма. Затем предложить учащимся вопросы для обсуждения, например: «Является ли это предписание алгоритмом?» Ответ необходимо обосновать.

В качестве примера рассмотрим разработанные нами шесть вариантов одного предписания, содержащего перечень действий по решению задачи на построение с помощью циркуля и линейки прямоугольника, одна из сторон которого равна длине данного отрезка АВ.

Вариант 1

1) отложить на прямой отрезок АВ;
2) поставить ножку циркуля в точку А;
3) установить раствор циркуля равным длине отрезка АВ;
4) провести окружность;
5) поставить ножку циркуля в точку пересечения окружности с прямой;
6) провести окружность;
7) поставить ножку циркуля в точку В;
8) провести окружность.

Этот вариант нельзя признать алгоритмом, поскольку перечень действий не завершен, т. е. не приводит к решению задачи.

Вариант 2

1) отложить на прямой отрезок АВ;
2) поставить ножку циркуля на отрезке АВ;
3) установить раствор циркуля равным длине отрезка АВ;
4) провести окружность;
5) поставить ножку циркуля в точку пересечения окружности с прямой;
6) провести окружность;
7) поставить ножку циркуля на отрезке АВ;
8) провести окружность;
9) последовательно соединить точки пересечения трех окружностей.

Этот вариант нельзя признать алгоритмом, поскольку в нем имеются непонятные команды (2, 7).

Вариант 3

1) отложить на прямой отрезок АВ;
2) поставить ножку циркуля в точку А;
3) установить раствор циркуля равным длине отрезка АВ;
4) провести дугу;
5) поставить ножку циркуля в точку пересечения дуги с прямой;
6) провести дугу;
7) поставить ножку циркуля в точку В;
8) провести дугу;
9) соединить точки пересечения трех дуг.

Этот вариант нельзя признать алгоритмом, поскольку в нем имеются неточные команды (4, 6, 8, 9).

Вариант 4

1) отложить на прямой отрезок АВ;
2) поставить ножку циркуля в точку В;
3) провести окружность;
4) поставить ножку циркуля в точку пересечения окружности с прямой;
5) провести окружность;
6) поставить ножку циркуля в точку А;
7) последовательно соединить точки пересечения трех окружностей прямыми.

Этот вариант нельзя признать алгоритмом, поскольку отсутствует преемственность указанных действий. Например, между командами 2 и 3 пропущена команда по установке величины раствора циркуля.

Вариант 5

1) отложить на прямой отрезок АВ;
2) поставить ножку циркуля в точку А;
3) провести окружность;
4) установить раствор циркуля равным длине отрезка АВ;
5) поставить ножку циркуля в точку пересечения окружности с прямой;
6) поставить ножку циркуля в точку В;
7) провести окружность;
8) провести окружность;
9) отметить четыре точки пересечения трех окружностей;
10) соединить эти четыре точки прямыми.

Этот вариант нельзя признать алгорит-

мом, поскольку в нем нет преемственности действий, в частности перепутан порядок команд 3 и 4.

Вариант 6

1) отложить на прямой отрезок АВ;
2) поставить ножку циркуля в точку А;
3) установить раствор циркуля равным длине отрезка АВ;
4) провести окружность;
5) отметить точку пересечения окружности с прямой буквой С;
6) поставить ножку циркуля в точку С;
7) провести окружность;
8) поставить ножку циркуля в точку В;
9) провести окружность;
10) отметить точки пересечения трех окружностей буквами Е, О, Р, Х.
11) последовательно соединить точки Е, О, Р, Х прямыми.

Этот вариант является алгоритмом, поскольку данный перечень понятных исполнителю преемственных действий, точно указывающих на отдельный завершенный шаг решения, обеспечивает выполнение предложенной задачи.

После того как суждение по каждому варианту предписания будет обосновано, можно переходить к третьему этапу работы с понятием «алгоритм» — преобразованию вариантов предписания. Так, если некоторое предписание признается алгоритмическим, то следует предложить переделать его в неалгоритмическое, и наоборот. Например, в варианте 3 нужно уточнить команды 4, 6, 8, 9, в варианте 4 — внести между командами 2 и 3 команду: «установить раствор циркуля равным длине отрезка АВ» и т. д.

Методически эти три этапа работы с понятием можно реализовать по-разному. Так, обсуждая разные виды предписаний в форме групповой дискуссии, учитель будет побуждать учеников к самостоятельным выводам и содержательным обоснованиям. Такую работу можно проводить и письменно в классе (в частности, составить проверочное задание), и в форме домашнего задания, которое будет способствовать самоконтролю при повторении усвоенного в классе материала.

Всесторонняя проработка содержания понятий позволит, на наш взгляд, значительно повысить качество их усвоения и будет содействовать развитию самостоятельного мышления учащихся. Кроме того, обсуждение содержания понятий и границ их применения способствует формированию у школьников устойчивой учебно-познавательной мотивации, необходимой для повышения их активности в учебном процессе.

Веселый урок

КАБИНЕТ ВТ

А. ИГЛИЦКИЙ

Школьная информатика: новый Вавилон?

Легенда утверждает, что строительство некоего архитектурного сооружения в Древнем Вавилоне не было успешным из-за того, что строители перестали понимать друг друга. Школьная информатика пока еще не достигла размеров (планировавшихся) вавилонской башни, однако смешение языков уже произошло. Существующие и перспективные ЭВМ для школы — БК-0010, «Агат», «Ямаха», «Корвет», УКНЦ — попарно несовместимы друг с другом[1]. Ситуация прискорбна; следует приложить усилия для ее исправления.

К сожалению, возможностей для изменений к лучшему мало. Разные машины имеют разные микропроцессоры, поэтому перенос программы в машинном коде заведомо невозможен[2]. Нужно обеспечить хотя бы перенос текстов программ (для последующей адаптации, ибо языки имеют индивидуальные особенности). Перенос может осуществляться только через магнитную ленту — единственный носитель информации, с которым может работать любая машина.

Прежде всего должен быть выработан единый стандарт на систему записи на магнитной ленте[3]. Не менее важен единый формат информации, позволяющий любой машине правильно прочитать запись, сделанную на другой.

Автор берет на себя смелость высказать некоторые предложения, которые могли бы лечь в основу быстрого, надежного и удобного для пользователей магнитофонного кода.

Физическая структура записи

1. Нулю соответствует прямоугольный импульс длительностью 200 мкс, единице — 400 мкс. Информация передается последовательностью битов (без разделителей между байтами).

2. Для предотвращения распространения ошибки запись разбивается на блоки длиной 2048 битов.

3. Каждый блок включает в себя контрольные биты, позволяющие обнаружить и исправить одну (как минимум) ошибку на блок.

Отметим, что при такой системе записи:

скорость передачи непостоянна — «все нули» передаются со скоростью 5000 бит/с, «все единицы» — 2500 бит/с. При равных долях нулей и елиниц средняя скорость передачи равна 3333,33 бит/с=416,66 байт/с;

за счет межблочных промежутков скорость передачи незначительно уменьшается,

[1] За исключением «Ямахи» и «Корвета», которые совместимы на уровне текстов программ, записанных на магнитной ленте. Это еще не обеспечивает переноса программ с одной машины на другую, так как языки программирования различны. Кроме того, вообще неясно, какова ценность совместимости перспективной отечественной машины с отнюдь не новой импортной, повторные закупки которой к тому же не ожидаются.

[2] Даже при одинаковых микропроцессорах совместимость не гарантируется — вспомним «Агат» и «Apple».

[3] Часто упоминаемый в литературе стандарт MSX имеет заметный недостаток — малую скорость передачи информации. При 1200 бит/с и трех дополнительных битах на каждый байт фактическая скорость передачи равна 109,09 байт/с. Для сравнения: у «Агата» в среднем около 175 байт/с.

практически можно рассчитывать на 400 байт/с;

наличие контрольных битов позволяет исправлять ошибки при считывании (или хотя бы часть их). Неисправимый сбой локализуется с точностью до блока, так что можно попытаться исправить ошибку повторным чтением того же блока (разумеется, это уже требует вмешательства пользователя).

Логическая структура записи

1. Байт 1 — тип записи. Например, А — ассемблер, В — Бейсик и т. п.
2. Байт 2 — длина имени файла (К).
3. Байты 3, ... К+2 — имя файла.
4. Байты от К+3 и до конца файла — собственно информация.

Мало-мальски компетентный читатель, разумеется, заметил, что в этих предложениях нет ничего нового — и не могло быть: все мыслимые коды магнитных лент были предложены и реализованы много лет назад. Автор взялся за перо с другой целью — обратить внимание на заметную помеху школьной информатике, в сущности, сводящуюся к мелкому техническому вопросу. Будем надеяться, что эта заметка ускорит его решение.

От редакции. Кому-то это предложение не нравится? Есть другие? Присылайте! Хотя... Сейчас, пожалуй, важнее не столько найти наилучший метод переноса текстов программ, сколько реализовать хоть какой-нибудь.

Учитесь отдыхать

Комплекс упражнений для профилактики зрительного утомления

ВАРИАНТ I (длительность 3—5 мин). Выполняется легко, без напряжения, сидя в удобной позе, позвоночник прямой, глаза открыты, взгляд устремлен прямо.

1. Взгляд направить влево—прямо, вправо—прямо, вверх—прямо, вниз—прямо, без задержки в отведенном положении. Повторять от 1 до 10 раз.

2. Взгляд смещать по диагонали: влево—вниз—прямо, вправо—вверх—прямо, вправо—вниз—прямо, влево—вверх—прямо и постепенно увеличивать задержки в отведенном положении, дыхание произвольное, но следить, чтобы не было задержки дыхания. Повторять 1—10 раз.

3. Круговые движения глаз: от 1 до 10 кругов влево и вправо. Вначале быстрее, потом как можно медленнее.

4. Изменение фокусного расстояния: смотреть на кончик носа, затем вдаль. Повторить несколько раз.

5. Смотреть на кончик пальца или карандаша, удерживаемого на расстоянии 30 см от глаз, а затем вдаль. Повторить несколько раз.

Смотреть прямо перед собой пристально и неподвижно, стараясь видеть более ясно, затем моргнуть несколько раз.

Сжать веки, затем моргнуть несколько раз.

6. Массировать веки глаз, мягко поглаживая их указательным и средним пальцами в направлении от носа к вискам. Или: глаза закрыть и подушечками ладони, очень нежно касаясь, проводить по верхним векам от висков к переносице и обратно, всего 10 раз в среднем темпе (выполнять чистыми руками).

7. Потереть ладони друг о друга и легко, без усилий, прикрыть ими предварительно закрытые глаза, чтобы полностью загородить их от света (на 1 мин). Представить погружение в полную темноту. Открыть глаза.

Вариант II (длительность 1—2 мин).

1. При счете 1—2 фиксация глаз на близком (на расстоянии 15—20 см) объекте, при счете 3—7 взгляд переводится на дальний объект, при счете 8 взгляд снова переводится на ближний объект.

2. При неподвижной голове на счет 1 поворот глаз по вертикали вверх, при счете 2 — вниз, затем снова вверх. Повторить 15—20 раз.

3. Закрыть глаза на 10—15 с, открыть и проделать движения глазами вправо и влево, затем вверх и вниз (5 раз), проделать несколько круговых движений глазами справа-налево и обратно (5 раз). Сво-

* Методические рекомендации по снижению зрительного утомления операторов, работающих с телевизионными индикаторами. Л., 1985. С. 27—30.

бодно без напряжения направить взгляд вдаль.

Вариант III (длительность 2—2,5 мин). Упражнения выполняются на рабочем месте сидя, откинувшись на спинку стула.

1. Смотреть прямо перед собой в течение 2—3 с, затем на 3—4 с опустить глаза вниз. Повторить упражнение в течение 30 с.

2. Поднять глаза кверху, опустить их книзу, отвести глаза в правую сторону, отвести глаза в левую сторону, повторить 3—4 раза (длительность 8 с).

3. Поднять глаза кверху, сделать ими круговые движения по часовой стрелке, сделать ими движения против часовой стрелки. Повторить 3—4 раза (длительность 15 с).

4. Крепко зажмурить глаза на 3—5 с, открыть на 3—5 с. Повторить 4—5 раз (длительность 30—50 с.).

Комплекс физических упражнений для снятия утомления

Данный комплекс может быть рекомендован для старшеклассников, работающих с ЭВТ. Количество упражнений и число повторений отдельных упражнений может быть скорректировано.

Потягивание

Исходное положение (и. п.) — сидя на стуле, ноги, согнутые в коленях, поставить на пол, руки на поясе.

1. Руки за голову, пальцы переплетены, прогнуться, голову наклонить назад, напрячь все мышцы тела — вдох.

2. Слегка наклониться вперед — выдох. Повторить 4—5 раз в среднем темпе.

Наклоны туловища с поворотом

И. п. — руки к плечам, ноги, согнутые в коленях, поставить на пол.

1. Наклониться вперед, правым локтем коснуться колена левой ноги — выдох.

2. И. п. — вдох.

Повторить 4—5 раз в медленном темпе.

Наклоны туловища в стороны

И. п. — сидя на стуле, руки на поясе.

1. Наклонить туловище влево, голову повернуть налево, правую руку за голову — выдох.

2. И. п. — вдох.

3. Наклонить туловище вправо, голову повернуть направо, левую руку за голову — выдох.

4. И. п. — вдох.

Повторить в медленном темпе 3—4 раза.

Расслабление мышц

И. п. — сидя на стуле, ноги вытянуть и поставить пятками на пол — вдох.

1—3. Последовательно расслаблять мышцы рук, туловища, ног — выдох.

4. И. п. — вдох.

Повторить 3—4 раза в медленном темпе.

Упражнения для пальцев

И. п. — сидя на стуле, руки согнуты в локтях, пальцы в кулаках.

1. Разжать пальцы — без напряжения.

2. Сжать пальцы в кулак — без напряжения.

3. Повторить в среднем темпе, дыхание ровное.

Перегруппировка веса тела

И. п. — рабочая поза.

1. Перегруппировать вес тела на левую часть — выдох.

2. И. п. — расслабить мышцы — вдох.

3. Перегруппировать вес тела на правую часть — выдох.

4. И. п. — расслабить мышцы — вдох.

Повторить 3—4 раза в медленном темпе, дыхание глубокое.

Упражнения для шеи

1. Влево—вправо вращение головой (быстро).

2. Вперед—назад (умеренный темп).

3. Вращение с максимальным расслаблением мышц шеи (голова «катится» по корпусу) сначала в левую, затем в правую сторону (очень медленно, не более 2 раз), мышцы лица расслаблены.

В журнале «Информатика и образование» (№ 3, 1987 г.) опубликована статья К. Шоломия «Построение обучающей программы». Автор считает необходимым внести следующие уточнения. **На с. 59** (правая колонка, первый абз. сверху) после слов «химия и т. п.» следует читать: ..., где операцию выбора формулы при решении задачи можно рассматривать как распознавание последней, т. е. как отнесение ее к классу задач, решаемых по данной формуле. Правила, позволяющие осуществить распознавание, в этих предметах в явном виде обычно не даются, но использование в них формального аппарата позволяет по крайней мере в некоторых случаях эти правила выявить. **На с. 60** (левая колонка, первый абз. снизу) после слова «Существенно» следует читать: ..., что автоматизация этапа тренировки в большей степени повышает вероятность достижения цели обучения, чем автоматизация первого этапа, поскольку позволяет значительно увеличить количество решаемых учеником задач и тем самым — возможности диагностики дефектов усвоения, а также их устранения.

Комплект временной нормативно-технической и методической документации

Комплект временной нормативно-технической и методической документации (НТМД) на разрабатываемые педагогические программные средства (ППС) подготовлен НИИ информатики и вычислительной техники (ИВТ) АПН СССР, НИИ школьного оборудования и технических средств обучения (ШОТСО) АПН СССР, НИИ физиологии детей и подростков (ФДиП) АПН СССР, НИИ педагогики МП УССР.

Комплект НТМД состоит из пяти руководящих методических документов (РМД):

РМД-1 Общие требования.

РМД-2 Виды и комплектность документов.

РМД-3 Техническое задание. Требования к содержанию и оформлению.

РМД-4 Сценарий. Требования к содержанию и оформлению.

РМД-5 Руководство учителя. Требования к содержанию и оформлению.

Комплект НТМД обсуждался на бюро Отделения дидактики и частных методик АПН СССР, координационном совещании разработчиков ППС в Новосибирске, на III Всесоюзном семинаре АН СССР «Разработка и применение программных средств ПЭВМ в учебном процессе» в Пущино, на школе молодых ученых «ЭВТ в учебном процессе» в Нарве, на Всесоюзной научно-практической конференции «Методика разработки ППС» в Новосибирске.

Временный комплект откорректирован НИИ ИВТ АПН СССР по результатам обсуждений и рекомендован комиссией по педагогическим программным средствам при Минпросе СССР для использования при разработке ППС. НИИ ИВТ АПН СССР ведет работу над новой редакцией комплекта.

Замечания и предложения к комплекту НТМД можно направлять по адресу: 630098, Новосибирск-98, ул. Приморская, 22, НИИ ИВТ АПН СССР.

Настоящая редакция временного комплекта вводится в действие до 01.01.89 г.

Руководящий методический документ РМД-1

Общие требования

Настоящий документ распространяется на программные средства (ПС), включающие педагогические программные средства (ППС) и инструментальные средства (ИС), разработанные и переданные в комиссию по приемке ППС, и устанавливает общие положения и требования к программным средствам.

1. Общие положения

1.1. Педагогические программные средства подразделяются на компьютерные курсы и отдельные ППС.

Компьютерный курс представляет собой учебный материал, включающий программы для ЭВМ и другие необходимые материалы для учителя и учащихся, позволяющие эффективно осуществлять изучение целостного учебного предмета.

Отдельные ППС представляют собой учебный материал, включающий программы для ЭВМ и другие материалы для учителя и учащихся, позволяющие эффективно осуществлять процесс обучения в рамках отдельной темы или вопроса.

1.2. Под инструментальным средством понимается программа, предназначенная для создания ППС.

2. Требования к педагогическим программным средствам

2.1. ППС должно отвечать определенной системе требований, в которую входят:

педагогические требования (2.2);

технические требования (2.3);

физиолого-гигиенические требования (2.4);

требования к программной документации (см. РМД-2).

2.2. Педагогические требования включают:

обоснование выбора темы общеобразовательного предмета для организации обучения с помощью ППС;

методические требования.

2.2.1. Обоснование выбора темы общеобразовательного предмета для разработки ППС.

Выбор общеобразовательного предмета или отдельной темы, изучение которой предполагается проводить с помощью ППС, следует аргументировать педагогическими целями использования ППС на уроке, реализация которых либо возможна только с помощью ЭВМ, либо интенсифицирует процесс обучения.

2.2.2. Методические требования к ППС предполагают необходимость учитывать принципы дидактики, своеобразие и особенности конкретных учебных предметов, предусматривающих специфику предмета и понятийного аппарата изучаемой науки. Разработка ППС должна быть педагогически целесообразной.

Ожидаемый результат соответствующих действий и операций (освоение знаний, умений и навыков, повышение уровня развития учащихся, сокращение учебного времени на освоение материала, повышение глубины и прочности знаний и т. д.) должен быть сформулирован в соответствии с тем или иным критерием, допускающим операциональную проверку.

2.2.3. Разработчики ППС должны продемонстрировать достижение сформулированных при создании ППС педагогических целей с помощью подготавливаемых или операциональных средств проверки.

2.3. Технические требования включают в себя:

---

* Подготовлен Министерством просвещения СССР и Академией педагогических наук СССР.

ориентацию ППС на массовые образцы КУВТ;
надежность и устойчивость работы ППС;
строгое соответствие функционирования ППС описанию в эксплуатационной документации;
восстановление системной области перед завершением работы программы (восстановление функциональных ключей, размеров экрана, цветов, отключение сети, принтера и т. д.);
минимальное время реакции;
обеспечение защиты от несанкционированных действий пользователя;
эффективное использование технических ресурсов, в том числе внешней памяти на магнитных носителях (для рабочих вводных или выводных файлов).

2.3.1. Требования устойчивости.

ППС должно обладать высокой устойчивостью к ошибочным или некорректным действиям учащихся и преподавателя, т. е.:

при работе ППС никакие действия учащихся и непреднамеренные действия преподавателя не должны приводить к блокированию работы какого-нибудь рабочего места или КУВТ в целом;

при работе по программе любые действия пользователя, не предусмотренные в данной программе, не должны влиять на ее работу;

в случае ошибок учащегося, преподавателя или ошибок в программе, препятствующих продолжению работы с рабочим местом или КУВТ, время приведения системы в рабочее состояние не должно превышать 10 с, при этом результаты предшествующей работы не должны теряться.

2.4. Физиолого-гигиенические требования устанавливают рекомендации к:
изображению информации;
разборчивости изображения;
эффективности считывания информации;
изображению знаковой информации;
цветовому изображению;
точности зрительного восприятия;
расположению текста на экране;
режиму работы.

2.4.1.1. Эффективность восприятия зрительной информации с экрана видеотерминала зависит от:
освещенности поля зрения, яркости фонового свечения экрана;
контрастности изображения на экране (разница в яркости между рассматриваемым изображением и фоном должна составлять, как правило, около 80 %);
достаточного размера изображения на экране и времени его экспозиции.

2.4.1.2. Разборчивость изображения зависит от формы и размеров частей изображения. Легче различаются и распознаются простые геометрические фигуры, буквы и цифры, составленные из прямых линий, чем фигуры, имеющие кривизну и много углов. В целях повышения эффективности восприятия алфавитно-цифровой информации рекомендуется использовать специально модифицированные конфигурации цифр и шрифтов.

2.4.1.3. Эффективность считывания цифр и букв определяется их угловыми размерами и яркостью свечения.

При больших угловых размерах (23—46 угл. мин) и высоких уровнях яркости (29—140 нт) способы начертания цифр и букв не отражаются на скорости и точности их воспроизведения.

При средних угловых величинах знака (6—12 угл. мин), но высоких уровнях яркости лучше различаются буквы и цифры с меньшей шириной штриха, а при больших угловых размерах знаков и низких уровнях яркости (0,5—6 нт) лучше различаются знаки с большой шириной штриха.

2.4.1.4. Для обеспечения хорошей читаемости цифр и букв рекомендуется придерживаться оптимальных соотношений основных параметров знака: высота и ширина знака, ширина штриха (толщина линии обводки). Рекомендуется использовать следующие оценки:

высота знака должна быть не менее 3,5 мм. Допустимое отношение ширины цифры и буквы к их высоте должно составлять 60—80 %, т. е. ширина знака должна быть не менее 2,1—2,8 мм;

толщина линии обводки знака должна составлять:

для обратного контраста — 10—15 % от его высоты (0,30—0,40 мм);

для прямого контраста — 12—15 % (0,30—0,40 мм).

Расстояние между знаками в строке должно составлять не менее 20 % от высоты знака, т. е. 0,7 мм. Расстояние между строками должно составлять не менее 50 % от высоты знака, т. е. 1,75 мм.

2.4.1.5. При цветовом изображении буквенно-цифровой информации на экране видеотерминала цветные знаки должны располагаться на светлом (светло-белом и светло-сером) или темном фоне. При цветовых изображениях, во избежание ошибки зрительного восприятия цвета, применять цветовой фон следует с осторожностью. Необходимо обеспечить использование таких сочетаний цветов, которые в монохромном режиме ученической ЭВМ не приводят к слиянию фрагментов изображения или нечитаемости текста.

2.4.1.6. Скорость и точнось зрительных восприятий связаны с попаданием средств отображения информации в разные зрительные зоны:

Зона I — с охватом +15 градусов от центральной плоскости (от линии взора) — является зоной оптимального рабочего пространства. Желательно, чтобы средства отображения информации попадали в эту зону.

Зона II — с охватом до +30 градусов от центральной плоскости; предназначена для менее точного и быстрого считывания информации.

2.4.1.7. Текст на экране видеотерминала не должен быть сплошным, покрывающим все поле экрана. Тексту должна придаваться форма, удобная для поиска его отдельных фрагментов. Структурная форма подачи информации будет способствовать выявлению логики материала, уменьшению количества одновременно предъявляемой информации, созданию благоприятных условий для работы.

2.4.1.8. При разработке ППС следует учитывать методические рекомендации, согласованные с Минздравом СССР (до 30.12.88 г. действуют «Гигиенические условия организации учебных

занятий с применением компьютеров в средней общеобразовательной школе. Временные методические рекомендации», Минздрав СССР, Москва — 1987 г.).

3. Требования к инструментальным средствам

3.1. Требования к инструментальным средствам устанавливаются в техническом задании на конкретное ИС (РМД-3).

Руководящий методический документ РМД-2

Виды и комплектность документов

1. Настоящий документ распространяется на педагогические программные средства и инструментальные средства, предназначенные для создания ППС.

2. Виды документов и их содержание приведены в табл. 1.

Таблица 1

Вид документа	Содержание документа
Техническое задание (РМД-3)	Наименование, область применения ППС, описание педагогических целей разработки ППС и средств оценки их достижения, требования к ППС, необходимые стадии и сроки разработки, испытаний, состав документации
Сценарий (РМД-4)	Цели обучения, система знаний, умений и навыков, необходимых для достижения поставленных целей, педагогические приемы достижения целей, структурно-функциональная схема и текст сценария, система средств контроля оценки достижения целей обучения
Текст программы с титульным листом	Запись программы с комментариями. На титульный лист выносится название ПС, авторы разработки, тип ЭВМ, объем занимаемой памяти, версия и дата разработки
Описание программы	Сведения о логической структуре и функционировании программы
Руководство программиста	Сведения, необходимые для работы с программой
Руководство системного программиста	Основные характеристики программы, комплектность, сведения для проверки, обеспечения функционирования, настройки программы на условия конкретного применения
Спецификация	Состав программы и документация на нее
Руководство учителя (РМД-5)	Сведения о месте и роли предлагаемой ППС, материалы, обеспечивающие работу учителя с ППС, в учебном процессе, средства оценки достижения педагогических целей, принятых при создании ППС, рекомендации по режимам работы, собираемой статистики. Методика использования ППС в учебном процессе

3. Комплектность документов, разрабатываемых на отдельные ППС, компьютерные курсы и инструментальные средства приведены в табл. 2.

Таблица 2

Вид документа	Вид программного средства: Отдельные ППС	Компьютерные курсы	Инструментальные средства
Техническое задание	*	*	*
Сценарий	*	*	—
Текст программы	+	+	+
Описание программы	—	—	+
Руководство программиста	—	—	+
Руководство системного программиста	—	—	+
Спецификация	—	+	+
Руководство учителя	+	+	—

При сдаче в комиссию по приемке ППС:
+ наличие документа обязательно
— наличие документа не обязательно
* в случае инициативной разработки документ может отсутствовать, в случае плановой — обязателен.

4. Выбор конкретного состава документов осуществляется на стадии технического задания. В комплект документов в зависимости от особенностей ПС разрешается включать и другие документы помимо перечисленных.

5. В случае, если одни и те же сведения необходимо привести в нескольких документах, эти сведения допускается приводить в одном из документов, поставляемых пользователю, а в других пользовательских докуметах делать соответствующие ссылки.

6. Оформление документов — по ГОСТ 19.105—78 и ГОСТ 19.106—78 (стандарты ЕСПД).

Руководящий методический документ РМД-3

Техническое задание. Требования к содержанию и оформлению

Настоящий документ устанавливает содержание и порядок оформления технического задания на разработку педагогических программных и инструментальных средств, создаваемых в учреждениях и организациях, независимо от ведомственной принадлежности, и подлежащих передаче в Отраслевой фонд алгоритмов и программ (ОФАП ППС).

1. Содержание

1.1. Техническое задание должно содержать следующие разделы:

введение;

основания для разработки;

назначение разработки;

требования к ПС;

средства оценки достижения педагогических целей (в случае ППС);

рабочий план (в случае плановой разработки).

В ТЗ допускается включать приложения. В зависимости от особенностей программы допускается уточнять содержание разделов, вводить новые разделы или объединять отдельные из них.

1.2. В разделе «Введение» указываются наименование и краткая характеристика области применения ПС.

1.3. В разделе «Основания для разработки» должны быть указаны:

полное наименование директивных документов, на основании которых создают ПС, организация, утвердившая эти документы, и дата их утверждения;

наименование и (или) условное обозначение темы, в рамках которой разрабатывается ПС, или указание, что разработка является инициативной.

1.4. Раздел «Назначение разработки» содержит подразделы: «Назначение разработки ППС» или «Назначение разработки ИС».

1.4.1. В подразделе «Назначение разработки ППС» указывают:

обоснование педагогической целесообразности создания ППС;

учебно-педагогические цели, которые должны быть достигнуты за счет применения ППС (объем знаний, навыков, умений);

характеристику категорий обучаемых, для которых предназначена программа;

возможность развития ППС и взаимосвязи с другими учебными программами.

1.4.2. В разделе «Наименование разработки ИС» указывают:

назначение ИС;

возможности ИС.

1.5. Раздел «Требования к ПС» должен содержать следующие подразделы:

требования к функциональным характеристикам;

требования к надежности;

требования к составу и параметрам технических средств.

1.5.1. Подраздел «Требования к функциональным характеристикам» может содержать следующие пункты:

описание структурных составляющих ПС;

принципы организации входных и выходных данных (сведения о формах обмена информацией между машиной и пользователями);

описание режимов работы ПС (например, для ППС при необходимости — возможность генерации индивидуальных заданий, сведения о возможных режимах работы ППС, уровнях сложности и подробности излагаемого материала, критериях выбора уровня сложности, подробности);

возможность сбора и ведения статистики (описание организации сбора, хранения и использования статистических показателей, получаемых в процессе работы).

1.5.2. В подразделе «Требования к надежности» указываются требования к обеспечению надежного, устойчивого функционирования, контроль входной и выходной информации, время восстановления после отказа, защита от несанкционированного доступа.

1.5.3. Подраздел «Требования к составу и параметрам технических средств» включает необходимый состав технических средств с указанием их основных технических характеристик.

1.6. Раздел «Средства оценки достижения педагогических целей» должен содержать подразделы:

средства оценки;

ожидаемый педагогический эффект.

1.6.1. Подраздел «Средства оценки» содержит описание системы средств (тесты, контрольные работы и т. д.), позволяющих оценить степень достижения ожидаемого педагогического эффекта.

1.6.2. В подразделе «Ожидаемый педагогический эффект» должен содержаться прогноз повышения эффективности учебного процесса за счет возможного сокращения сроков обучения и (или) повышения прочности усвоения, освобождения учителя от рутинных операций, повышения интереса к обучению, создания условий для творческой деятельности и реализации инициативы учащихся.

1.7. В разделе «Рабочий план» устанавливают необходимые этапы, содержание работ, а также сроки разработки и определяют исполнителей.

2. Оформление

2.1. Техническое задание оформляют в соответствии со стандартами ЕСПД (ГОСТ 19.106—78, ГОСТ 19.104—78, 19.201—78).

Руководящий методический документ РМД-4

Сценарий. Требования к содержанию и оформлению

Настоящий документ устанавливает порядок построения и оформления программного документа «Сценарий».

Под сценарием ППС понимается описание сюжетной схемы урока, текстовой и графической информации, выводимой на экран дисплеев, порядка предъявления и количества выводимой информации, а также всех методических примеров, предлагаемых для использования в обучающей программе.

1. Содержание

1.1. Сценарий должен включать следующие разделы:

введение;

назначение и область применения;

содержание и методика обучения;

сценарный план.

1.2. В разделе «Введение» должны быть указаны наименование программы, тема разработки, организация-разработчик, адрес, фамилии и инициалы авторов.

1.3. Раздел «Назначение и область применения» должен содержать следующие подразделы:

предметная область ППС;

категория обучаемых, для которых предназначен ППС;

методические цели разработки, ожидаемый педагогический эффект, способ его оценки;

место и функции программы в структуре учебного процесса.

1.4. Раздел «Содержание и методика обучения» должен включать следующие подразделы:

исходный уровень обучаемых;

содержание обучения;

методика обучения;

средства контроля оценки достижения целей обучения.

1.4.1. Подраздел «Исходный уровень обучаемых» содержит характеристику предварительных знаний, умений и навыков, необходимых для успешной работы с программой.

1.4.2. Подраздел «Содержание обучения» включает описание системы знаний, умений и навыков, формирующихся в процессе работы с ППС.

1.4.3. Подраздел «Методика обучения» содержит описание способа формирования данной системы знаний, умений и навыков, методическую разработку занятий (тема, цели и задачи обучения, материальное обеспечение, место проведения, рассматриваемые вопросы, краткое содержание вопросов, методические приемы, хронометраж и т. д.).

1.4.4. Подраздел «Средства контроля» содержит описание средств, позволяющих оценить степень достижения ожидаемого педагогического эффекта (контрольные вопросы, тесты и т. п.).

1.5. Раздел «Сценарный план».

В этом разделе приводятся подробное описание структурно-функциональной схемы будущей программы, режимов ее работы, содержание и язык взаимодействия пользователя с ППС.

Раздел содержит подразделы:

структурно-функциональная схема;

описание кадров;

статистика.

1.5.1. Подраздел «Структурно-функциональная схема» содержит перечень и взаимосвязь кадров программы.

Целесообразно иллюстритовать материал данного раздела рисунками и таблицами.

1.5.2. Подраздел «Описание кадров» осуществляется по каждому шагу и содержит:

номер кадра в шаге;

длительность работы с ним;

описание зон ввода-вывода информации (их количество, размещение, назначение);

текстовое и графическое содержение кадра;

описание динамических эффектов;

особенности применения звука и цвета;

методические указания по действиям учителя и ученика;

взаимосвязь с другими подразделами данного шага и условия перехода к ним.

Описание кадров желательно оформить таблицей и схемой взаимосвязей (см. табл. 3).

1.5.3. Подраздел «Статистика» включается в сценарий по необходимости и содержит описание:

состава фиксируемых показателей;

кадров программы, в которых фиксируются показатели;

алгоритмов управления программой и обратной связи в зависимости от значений снимаемых показателей;

процедур обработки показателей.

2. Оформление

2.1. Сценарий оформляют в соответствии со стандартами ЕСПД (ГОСТ 19.104—78, ГОСТ 19.105—78, ГОСТ 19.106—78).

Руководящий методический материал РМД-5

Руководство учителя. Требования к содержанию и оформлению

Настоящий документ устанавливает требования к содержанию и оформлению эксплуатационного документа «Руководство учителя», определенного в РМД-2.

1. Содержание

1.1. Руководство учителя состоит из трех разделов:

введение;

порядок работы с ППС;

методические рекомендации по использованию ППС в учебном процессе.

1.2. Раздел «Введение» должен включать следующие подразделы:

общие сведения;

педагогические цели;

содержание ППС.

1.2.1. Подраздел «Общие сведения» включает:

наименование программы, организации-разработчика, фамилии и инициалы авторов;

указание места ППС в структуре предмета, курса, темы, урока;

указание категории обучаемых;

указание исходных знаний, умений и навыков, необходимых для начала работы с ППС.

1.2.2. Подраздел «Педагогические цели» включает:

указание целей применения ППС;

описание ожидаемого результата и методов проверки достижения этого результата.

1.2.3. Подраздел «Содержание ППС» включает:

описание содержания и формы подачи учебного материала;

обоснование выбранного содержания, т. е. объяснение, почему данное содержание ППС позволяет достигнуть поставленных целей;

указание вспомогательной литературы.

1.3. Раздел «Порядок работы с ППС» должен включать следующие подразделы:

описание среды функционирования;

действия по вызову и загрузке;

действия по управлению программой;

методика ознакомления с программой, самоучитель для учителя.

1.3.1. Подраздел «Описание среды функционирования» должен содержать сведения о типе и

Таблица 3

Описание кадров на шаге №_____

Наименование ППС

№ п/п	Длительность	Описание зон ввода-вывода информации на экране	Описание текстового содержания зоны	Описание динамич. эффекта (звук, цвет...)	Описание действий ученика	Взаимосвязь с другими кадрами № Условия перехода кадра
1.		зона ввода зона вывода постоянная зона			1. 2. 3. 4.	3. 7. 4.

конфигурации ЭВМ, операционной системе, языке программирования, дополнительном аппаратном или программном обеспечении, необходимом для работы программы.

1.3.2. Подраздел «Действия по вызову и загрузке» должен содержать всю необходимую информацию, позволяющую осуществить загрузку, запуск, выполнение и завершение программы неподготовленным пользователем.

1.3.3. Подраздел «Действия по управлению программой» включает описание:

языка пользователя (описание интерфейса пользователь — программа);

возможных режимов;

возможности повторного запуска, останова, продолжения, выхода из программы, повторного входа в нужный пользовательский кадр.

1.3.4. Подраздел «Методика ознакомления с программой» должен содержать материал, позволяющий учителю наиболее эффективным способом ознакомиться со всеми функциональными возможностями и содержанием ППС.

1.4. Раздел «Методические рекомендации по использованию ППС в учебном процессе» должен содержать подразделы:

методы и приемы организации учебной деятельности;

методы оценки учебной работы.

1.4.1. Подраздел «Методы и приемы организации учебной деятельности» содержит:

описание способа формирования системы знаний, умений и навыков, методическую разработку плана занятий (тема, цели и задачи обучения, учебное оборудование и режим его использования, рассматриваемые вопросы, краткое содержание вопросов, методические приемы и т. д.).

1.4.2. Подраздел «Методы оценки учебной работы» включает:

описание средств контроля уровня знаний учащегося в процессе работы с ППС;

описание средств достижения педагогических целей, принятых при создании ППС;

описание результатов экспериментальной проверки для сравнения с результатами практики использования в учебном процессе.

1.5. Разрешается дополнять документ разделами, не упомянутыми в настоящем руководстве, а также не приводить в документе часть упомянутых в настоящем руководстве сведений, если для разрабатываемого ППС они не имеют конкретного смысла. Отсутствие конкретной информации по подразделу в целом фиксируют в таком подразделе с необходимыми пояснениями.

2. Оформление

2.1. Руководство учителя оформляется в соответствии со стандартами ЕСПД (ГОСТ 19.104—78, ГОСТ 19.105—78, ГОСТ 19.106—78).

Разработчики публикуемого материала ждут от читателей журнала замечаний и предложений по подготовке следующей редакции НТМД.

Веселый урок

Ю. АСТРАТОВ,
учитель математики и информатики, школа № 719, Зеленоград

Размышления об использовании компьютера в учебном процессе

Листаю и перечитываю журнал «Информатика и образование» № 1 за 1987 г. Само название журнала говорит о том, что ЭВМ должны войти непосредственно в школьное образование. Хочу найти в этом журнале ответы, как помогают компьютеры нашему всеобщему среднему, но... информатики много, а вот об образовании, увы, так мало. А судя по тому интересу, который проявляет учительство страны к применению электронно-вычислительной техники в учебном процессе (в школу постоянно приезжают делегации со всех концов страны), учителя нашей школы считают, что молчать не имеем права. Особенно волнует этот вопрос те коллективы, которые смогли получить то или иное количество персональных компьютеров. Машины уже приходят в школы, а вот программное обеспечение...

С болью и горечью прочитали статью «Компьютер с акцентом» в «Комсомольской правде». Еще раз подтверждается мысль, что там, где образовалась пустота, обязательно найдутся «охотники» ее заполнить. Медленно, очень медленно настраиваются на компьютеризацию в школьном образовании АПН и все соответствующие НИИ.

В связи с этим хотелось бы поделиться тем небольшим опытом, который наша школа накопила за три года работы с классом персональных ЭВМ (ДВК-1 и ДВК-2М).

Как научить каждого и научить хорошо? Не всякий учитель может стать такой личностью, как Шаталов, Волков, Ильин, Амонашвили. И сколько бы ни было выпущено книг об их опыте, не все и не у каждого учителя получится так же.

Говоря об интенсификации учебно-воспитательного процесса, мы прежде всего затрагиваем, хотим этого или не хотим, труд учителя, его нагрузку. Учителю приходится трудиться более напряженно, а всегда ли интенсивен труд ученика? Порой учителя достигают подлинных высот профессионализма, применяя и ТСО, и всевозможные методы и приемы, чтобы создать условия напряженной работы учащихся. Особенно удачно проходят уроки-лекции, диспуты, уроки-сообщения, лабораторные и практические занятия, но гораздо тяжелее идут уроки, где вырабатываются устойчивые и прочные навыки. Здесь нам порой очень трудно интенсифицировать процесс обучения, потому что получение навыков зависит от труда того, кто эти навыки должен получить. И вот тут «красивые», блестящие уроки подчас теряют смысл, ибо если ученики не привыкли к самостоятельному труду, то к урокам они относятся, как к просмотру телепередач: интересно было или нет. Ведь не секрет, что мы можем встретить ученика, окончившего восьмилетнюю школу, с плохим знанием таблицы умножения, ученика, который для простейших расчетов хватается за бумагу, чтобы произвести действия «в столбик», или вынимает калькулятор, чтобы произвести элементарные вычисления. Вся система знаний опирается на те основные навыки и понятия, над которыми ученик не должен задумываться, ответы на которые должны выдаваться автоматически, как только в этом возникает необходимость. Это фундамент, и он должен быть незыблем, иначе, какое бы красивое здание мы ни построили, оно рухнет. Сила метода Шаталова в том, что он смог создать жесткую систему для выработки основ знаний, а ведь каждый учитель размышляет: «Давайте мне учеников с прочными навыками, а пользоваться

этими навыками, «творить» я их научу». Именно в отсутствии прочных навыков у абитуриентов который год справедливо упрекает нас высшая школа. Наверное, поэтому поступление в вуз через «репетиторство» стало массовым явлением.

Какой же путь избрали мы? Во втором номере журнала «Информатика и образование» за 1986 г. в статье завуча нашей школы Л. А. Филимоновой было рассказано об опыте создания учебных и обучающих программ учителями и учащимися школы, о том большом интересе, который проявили ученики I—VIII классов к работе с этими программами. Это были контролирующие программы, демонстрационные, информационно-справочные, тренажеры. Существует мнение, что основная задача применения ЭВМ в обучении — это осуществление контроля за знаниями учащихся, а отсюда и основной вид программ — контролирующие программы. Но мы считаем, что чистый, формальный контроль учителю не трудно провести и самому, но увидеть ошибку, проанализировать ее и помочь исправить значительно более важно, чем поставить оценку. Не отвергая ни один вид программ для школы, о которых говорилось выше, мы пришли к выводу о том, что более всего на данном этапе потребностям школы удовлетворяет вид программы, который мы назвали «обучающий тренажер», так как именно он становится непосредственным помощником учителя, помогает ему осуществить дифференцированный подход к каждому ученику, вовремя заметить пробелы в знаниях учащихся и устранить их. В чем суть такого вида программ, я попробую объяснить на примере одной обучающей программы-тренажера по математике «Сложение обыкновенных дробей».

Программа построена по методике школьного урока. Состоит из 3 частей:

сложение дробей, один из знаменателей которых кратен другому;

знаменатели дробей — взаимно простые числа;

знаменатели дробей имеют одинаковые делители.

Каждый из этих разделов может работать в автономном режиме, хотя все три позволяют выработать общее правило сложения дробей.

Начало каждой части состоит из объяснения машиной на примере конкретной задачи сложения дробей одного из перечисленных видов. Символы, записи появляются на экране дисплея медленно, с той скоростью, с какой ученик мог бы сам писать в тетради. Машина «объясняет» так, как бы объяснял учитель. Если ученику понятно объяснение, то машина предлагает ему потренироваться вместе с ней (в противном случае ЭВМ готова повторить объяснение). Она предлагает ученику постепенно заполнять пропуски (мигая знаком вопроса), причем в той последовательности, которая необходима для осознанного восприятия (сначала дополнительный множитель для первого знаменателя, потом для числителя, затем та же последовательность для другой дроби). Если ученик ошибся, машина не допускает его к дальнейшей работе, а сразу указывает на ошибку, объясняет, почему и что надо написать, и исправляет ее, приглашая ученика далее продолжить решение, весело подмигивая знаком вопроса.

Машина «не разрешит» дальнейшую работу до тех пор, пока все пропуски в примере он не заполнит правильно, иными словами, весь пример не решит сам (если была допущена ошибка, машина предложит ему аналогичный пример).

Если ученик ошибался более чем в трех примерах, машина возвращает его к началу обучения. Если же упражнение выполнено правильно, то ЭВМ указывает на общие шаги, которые пришлось выполнить при решении примера данного типа, тем самым подготавливая учащегося к выводу общего правила.

По этому же типу созданы и две другие части программы, а в итоге вырабатывается общее правило сложения дробей с разными знаменателями. Это обучающая часть программы. Но программа позволяет работать и в режиме тренажера, и в режиме контроля. На тренажере хочу остановиться особо, ибо эту часть программы мы считаем обучающим тренажером.

ЭВМ случайным образом генерирует две правильные дроби с разными знаменателями и предоставляет учащемуся возможность найти их сумму на основе выведенного правила. После выполнения примера учащимся ЭВМ указывает первую же встретившуюся у него ошибку (если таковая имеется), дает время подумать и предлагает выполнить этот же пример заново с того места, где она была допущена, оставляя на экране дисплея только ту часть примера, которая выполнена правильно. В случае, если учащийся не смог дать правильный ответ второй раз в этом же месте, ЭВМ дает анализ характера ошибки и указывает пути для нахождения правильного ответа. Если и после этого учащийся ошибся там же, то дается полное правильное решение всего примера.

В итоге, работая с этой программой, учащийся просто не сможет не научиться складывать дроби.

При работе с программой в режиме контроля учащийся имеет возможность получить оценку за выполненную работу, хотя повторяю, что получение оценки не является главной целью.

При составлении программ мы стараемся, чтобы не происходило наложение нескольких трудных тем и отрабатываем одну. Мы идем от простого, а не наоборот. Простое отрабатываем при помощи ЭВМ до автоматизма, а сложному учим на уроке, т. е. творчество и учителя и ученика переносим на урок. Так, например, в программе-тренажере «Формулы сокращенного умножения» мы не даем для машины сложных выражений не потому, что не смогли бы этого сделать, а чтобы не усугублять трудности освоения формул трудностями применения свойств степени. Таким образом, машина не заменяет учителя, а помогает ему. Она только стимулирует работу учащихся на уроке, в коллективе, потому что навыки есть теперь у всех!

Применение обучающей программы-тренажера «Решение приведенных квадратных уравнений» с применением теоремы Виета в VII классе позволило учащимся, будучи в VIII классе, опираясь на полученные навыки, свободно оперировать с разложением квадратных трехчленов на множители, помогло в решении квадратных неравенств, в применении метода интервалов. Эти же навыки дали возможность учителю рассмотреть во время урока искусственные способы решения систем уравнений с использованием теоремы Виета, а таких способов, как известно, много, знание их повысило математическую культуру учащихся.

Насколько же прочны навыки, приобретенные с помощью ЭВМ? В этом учебном году учащимся девятых классов без предупреждения предложили выполнить срезовую работу по теме «Применение теоремы Виета». Было представлено 35 вариантов, время выполнения — 10 мин. Содержала она 8 заданий следующего типа:

1. Решить уравнения:

$x^2-13x+40=0$ $x^2+x-6=0$

$x^2+11x+30=0$ $y^2-2y-35=0$

$0{,}5x^2+5x+8=0$

2. Решить системы уравнений:

1) $\begin{cases} x-y=3 \\ xy=-2 \end{cases}$ 2) $\begin{cases} x^2+y^2=25 \\ xy=12 \end{cases}$

3. Найти меньший корень квадратного уравнения:

$2x+px+q=0$, если $p=-5$, $q=-6$

Среди учащихся девятых классов было выделено две равноценные по успеваемости группы: те, которые работали с программой «Решение приведенных квадратных уравнений», и те, которые с этой программой не работали.

В первой группе все справились с решением квадратных уравнений, половина учащихся справились с заданием полностью, а лучшие сдали работу через 5—7 мин. Из второй группы учащихся ни один полностью не справился со всей работой, никто не находил корни уравнения «в уме», а к системам приступали только с помощью подстановки.

Нами была проведена еще одна срезовая работа по математике в IV классе на решение линейных уравнений. Этим школьникам, которые, будучи во II и III классах, еженедельно ходили в класс ДВК «работать» по программам-тренажерам (табличное и внетабличное умножение и деление, приемы устного счета, нахождение неизвестных компонентов, увеличить-уменьшить «на» и «в» и т. д.), предложили без «прохождения» этой темы в классе поработать с программой «Решение линейных уравнений». Затем им была предложена работа в 40 различных вариантах на решение 5 линейных уравнений. Время ограничивалось 5 мин. Первые учащиеся стали сдавать работу через 1—3 мин, 24 человека выполнили работу безошибочно, 10 человек допустили одну ошибку, решил только три примера один человек, и один ученик не справился с работой. Характерно, что при выполнении работы в классе не было допущено ни одной ошибки на табличное и внетабличное деление! Мы считаем, что это результат работы с ЭВМ. А как нравится ребятам учиться с машинами! Наибольший интерес проявляют учащиеся I—VII классов, а не это ли период приобретения и усвоения прочных навыков? Имея в школе два компьютерных класса (по 12 персональных машин в каждом), мы выработали в школе и методику работы с программами. Таких видов работ четыре:

1. Учитель предусматривает во время урока работу всех учащихся за пультом ЭВМ. Он планирует свой урок так, чтобы приход учащихся в класс и уход из него не влиял на работу остальных, чтобы пришедшие смогли сразу «включиться» в работу (это может быть урок по повторению, индивидуальная или групповая консультация, работа учащихся с учебником и т. д.). Учитель находится в классе, а учащиеся группами уходят работать на ЭВМ. За работой в классе ДВК следит лаборант, он фиксирует результаты работы учащихся с машиной, которые затем передает учителю.

2. Учитель считает, что только 10—12 учащимся необходимо поработать с программой индивидуально во время урока.

Он направляет в кабинет ДВК эту группу, а сам с более сильными оставшимися учениками может рассмотреть новые приемы и методы решения задач, проследить за работой учащихся, проявляющих повышенный интерес к предмету.

3. После прохождения той или иной темы учитель может рекомендовать отдельным учащимся (или группе) после уроков в определенное время поработать с программой (обычно на таких занятиях присутствует сам учитель).

4. При ответе ученика, при проверке самостоятельной или контрольной работы учитель может обнаружить пробелы в знаниях ученика, которые он может ликвидировать, поработав самостоятельно с обучающей программой.

В этом случае учитель пишет направление в класс ДВК (как участковый врач дает талончик к врачу-специалисту), в котором лаборант согласовывает с учеником время и номер машины, на которой он может прийти позаниматься самостоятельно. После занятий лаборант на этом же направлении указывает результат работы.

Хотя в школе уже действенно работают около 70 различных программ, этого, конечно, мало. В школе программы могут писать только два учителя и группа учащихся, увлекшихся идеей создания учебных программ. Почему мы с некоторым недоверием относимся к профессионалам-программистам? У нас есть в школе пакет программ, разработанный сотрудниками НИИ школ Минпроса РСФСР и Московским институтом электронной техники. К сожалению, эти программы однотипны, и работа с ними «приедается» учащимся. Ведь не интересно на уроках и иностранного языка, и русского языка, и истории работать всегда одинаково! Обучающая программа должна быть, как и урок учителя, не похожа одна на другую, но каждая содержать тот богатейший опыт, то разнообразие и множество приемов и методов обучения, которые выработало наше учительство. Поэтому решающее слово при написании сценария программы принадлежит учителю. А учитель только тогда сможет научиться писать сценарии, когда он воочию увидит возможности ЭВМ, увидит различные типы программ.

К сожалению, у нас нет центра, где учителя могли бы ознакомиться с таковыми. Попытка представить на ВДНХ в 1986 г. два дисплейных класса, оснащенных ЭВМ типа ДВК и «Агат», не является решением этой серьезной проблемы. А ведь именно в сценариях так нуждаются сейчас программисты, готовые писать программы для школ, но не знающие «школьных болячек» и предлагающие использовать ЭВМ в качестве «контролера». Обучение написанию сценариев — это своего рода обучение учителей алгоритмическому мышлению.

Почему бы не использовать для показа различных учебных программ для ЭВМ телевидение? Почему бы не помещать сценарии обучающих программ в методических журналах? А может быть, объявить и конкурс по отдельным темам? Вот тогда прекратились бы, наверное, бесчисленные поездки учителей в поисках хоть каких-нибудь программ, негласное распространение программ, слабых в методическом отношении и далеко несовершенных.

ВНЕКЛАССНАЯ РАБОТА

Олимпиада по информатике: какой ей быть?

Олимпиады — это ученические праздники, состязания в сообразительности, умении быстро и точно применять свои знания. Для педагогов — это возможность проверить, насколько эффективно освоение предмета.

В школах страны состоялись первые городские и областные олимпиады по информатике. Обратившись к их содержанию, мы должны отметить, что многие из этих олимпиад точнее было бы назвать олимпиадами по программированию, где во главу угла ставится, как правило, умение программировать и часто в самом узком смысле этого понятия — умение писать тексты программ. На наш взгляд, это противоречит дидактическим задачам нового учебного предмета. Но дело в том, что рекомендуемые школам задания для олимпиад в подавляющем большинстве — задания по составлению программ, и только.

«Олимпиадному контролю» должны подвергаться все основные понятия курса информатики. Задания придумать непросто, но опыт придет. Искусственно ограничивать содержание столь важного предмета недопустимо. На олимпиаде полученные школьниками сведения и приобретенные умения должны проверяться в их совокупности.

Скорость становления нового предмета высока, и ошибки, допущенные на старте, могут иметь весьма нежелательные последствия.

В Крыму уже многие годы школьники имеют возможность изучать основы программирования и вычислительной техники. С 1963 г. в малой академии наук школьников Крыма «Искатель» систематически ведутся такие занятия. Накоплен значительный опыт, особенно во внеклассной работе. В нынешнем году будет проводиться уже XVII олимпиада по основам информатики и вычислительной техники.

Приведем несколько тем заданий, предлагаемых школьникам:

на различные системы счисления (рассматриваются и весьма экзотические системы: троичная уравновешенная, негадвоичная и др.);

на отыскание алгоритмов в различных предметных областях (рассматриваются алгоритмы построения фигур одним циркулем, алгоритмы игр и др.);

на простейшее кодирование и декодирование;

на разработку алгоритмов в необычных системах, например системе Поста, Маркова и Тьюринга;

на решение логических задач с последующим составлением программ их решения на ЭВМ;

на разработку простейших логических схем устройств (речь идет только о функциональных схемах, а не об электронных);

по программированию (с использованием различных языков). И др.

Параллельно с олимпиадой проводится конкурс творческих работ конструкторского, теоретического и реферативного характера. Это интересный элемент праздника знаний.

В. КАСАТКИН

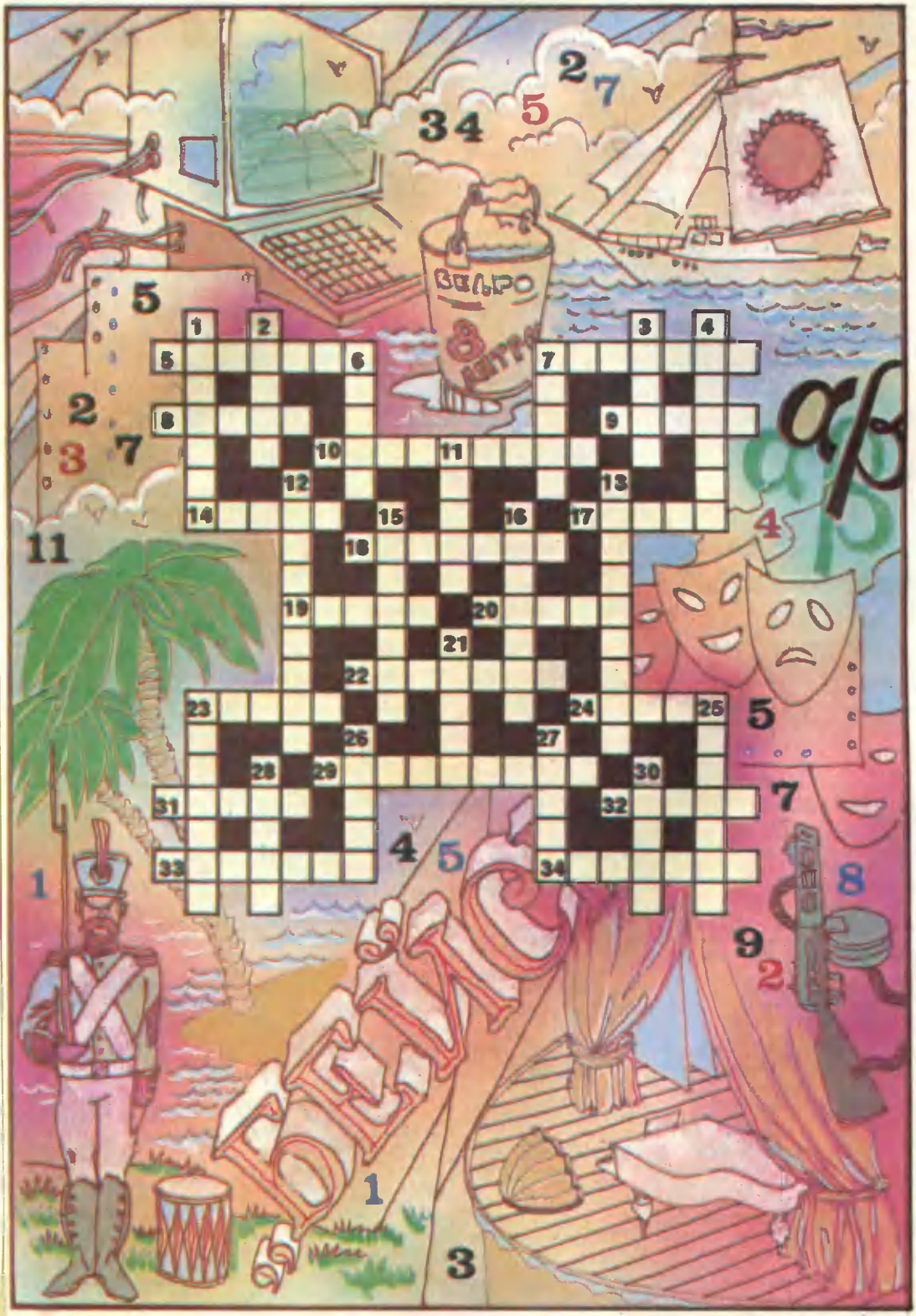
Кроссворд

По горизонтали: 5. Разработанная в США большая программа для аналитических вычислений на ЭВМ. 7. Устройство вывода информации на печать. 8. Знак алфавита языка программирования. 9. Морское путешествие. 10. Величина, не изменяющаяся в ходе работы программы. 14. Процесс нахождения в файле необходимой информации. 17. Площадка, на которой происходит театральное действие. 18. Важнейшая характеристика машинной памяти. 19. Специальное машинное слово, предназначенное для выделения в других словах определенных разрядов. 20. Один из наиболее распространенных алгоритмических языков. 22. Часто используемое название математического обеспечения ЭВМ. 23. Тропическое растение — каучуконос. 24. Известный французский киноактер, снимавшийся в фильмах «Пармская обитель», «Фанфан-Тюльпан», «Красное и черное». 29. Комплекс операций, производимых над информацией в ЭВМ. 31. Предложение, высказанное в устной форме. 32. Специальный знак, применяемый при необходимости передачи управления от одной части программы к другой. 33. Математическая модель устройства, перерабатывающего цифровую информацию. 34. Прямая, перпендикулярная к касательной в ее общей точке с кривой.

По вертикали: 1. Полупроводниковый прибор с управляемой емкостью. 2. Серия советских электронных клавишных вычислительных машин. 3. Автор книги «Искусственный интеллект». 4. Отрезок, соединяющий вершину треугольника с серединой противоположной стороны. 6. Химический элемент, инертный газ. 7. Приспособление для крепления и соединения электронных элементов. 11. Электронная лампа. 12. Программа для автоматического перевода с автокода на машинный язык. 13. Одна из операций ЭВМ. 15. Буква греческого алфавита. 16. Вертикальный ряд позиций для нанесения перфорации на перфокарту. 21. Импульс, используемый для выделения других импульсов. 23. Город в Болгарии, в котором проводятся фестивали юмора. 25. Алгоритмический язык высокого уровня. 26. Единица количества информации, равная 1024 К. 27. Устройство вывода дисплея. 28. Войсковое подразделение. 30. Один из создателей аналитической геометрии.

Кроссворд составлен ЭВМ по алгоритму

Е. Поезда и А. Поезда.


ВЕЛЬРО

БИБЛИОГРАФИЯ
(советская литература)

Библиография состоит из четырех разделов. В разделе I указана литература по общим вопросам внедрения ЭВМ в учебный процесс. Здесь, в частности, перечислены основные Положения о мерах по обеспечению компьютерной грамотности учащихся, о подготовке преподавателей к ведению нового школьного курса и др. Раздел II включает публикации, раскрывающие цели, содержание и структуру, а также методику преподавания курса «Основы информатики и вычислительной техники».

Рекомендуемая литература раздела III знакомит с вопросами использования ЭВМ как средства обучения. В публикациях обсуждаются подходы к решению таких проблем: выявление целей, места и границ применения ЭВМ в процессе обучения; отбор содержания, подлежащего усвоению с помощью компьютера, разработка методов обучения с учетом ЭВМ и т. п.

Раздел IV включает работы, посвященные психолого-философским аспектам компьютеризации обучения.

Библиография имеет нумерацию по разделам, материал внутри разделов расположен в алфавитном порядке. Большая часть указанных публикаций сопровождается краткой аннотацией.

Библиография не претендует на абсолютно полный охват литературы по рассматриваемой проблеме. Мы адресуем читателей к указателям, приведенным в конце раздела I.

Публикация подготовлена Т. А. Сергеевой и О. А. Дяшкиной (отдел информатики и вычислительной техники НИИ СиМО АПН СССР, зав. лаб. А. А. Кузнецов). Ответственность за точность библиографических сведений несут авторы.

Раздел I

1. Вторая грамотность: О мерах по внедрению ЭВМ в учебный процесс, создании научного редакционного совета из крупных ученых — специалистов в этой области // Учит. газета. 1985. 15 янв.

2. *Гельтищева Е.* Гигиена труда в КВТ // Информатика и образование. 1986. № 1. С. 80—83.

3. *Гершунский Б. С.* Прогностический подход к компьютеризации // Советская педагогика. 1986. № 7. С. 43—49.

Статья освещает теоретические и опытно-экспериментальные исследования по основным направлениям использования компьютерной техники.

4. *Гершунский Б. С.* Теоретико-методологические основания компьютеризации в сфере образования: (Прогностический аспект) // Обзор информации НИИОП АПН СССР: Обзоры по информационному обеспечению целевых комплексных научно-педагогических программ по решению важнейших научных проблем. М., 1985. Вып. 5(15).

5. Основы информатики и вычислительной техники: Программа для средних учебных заведений // Среднее спец. образование. 1985. № 9. С. 25—27.

6. Основы информатики и вычислительной техники: Пробное учебное пособие для средних учебных заведений. Программа / Под ред. А. П. Ершова, В. М. Монахова. М., 1985. Ч. 1.

7. Изучение основ информатики и вычислительной техники: Пособие для учителя / Под ред. А. П. Ершова, В. М. Монахова. М., 1985.

8. Кабинет вычислительной техники всех типов средних учебных заведений (на базе персональных микро-ЭВМ): Методические рекомендации // Информатика и образование. 1986. № 3. С. 5—18.

Статья содержит конкретные рекомендации по оснащению КВТ учебным оборудованием, учебно-наглядными пособиями, оргтехникой и мебелью, включает требования по оборудованию рабочих мест учителя и учащихся, а также гигиенические требования к организации КВТ.

9. Компьютер в обучении: Психолого-педагогические проблемы («круглый стол») // Информатика и образование. 1986. № 5. С. 65—90.

10. Научно-методические основы информатики и вычислительной техники: Программа для подготовки учителей // Бюл. Министерства высшего и среднего спец. образования СССР. 1985. № 8. С. 30—38.

11. О введении в действие перечней технических средств, учебно-наглядных пособий и мебели для кабинетов вычислительной техники: Приказ от 30 авг. 1985. // Бюл. нормативных актов Министерства просвещения СССР. 1986. № 1. С. 8—11.

12. О введении в кабинетах вычислительной техники институтов усовершенствования учителей должности методиста по использованию электронно-вычислительной техники: Приказ от 13 дек. 1985 г. № 220 // Бюл. нормативных актов Министерства просвещения СССР. 1986. № 5. С. 32.

13. О мерах по обеспечению компьютерной грамотности учащихся сред. спец. учебных заведений и широкого внедрения электронно-вычислительной техники в учебный процесс // Бюл. Министерства высшего и среднего спец. образования СССР. 1985 № 7. С. 1—4.

14. О подготовке руководящих и педагогических кадров народного образования по основам информатики и вычислительной техники // Информатика и образование. 1986. № 2. С. 7—11.

Представлены тематический план и программа подготовки работников народного образования по основам информатики и вычислительной техники.

15. О подготовке студентов физико-математических факультетов пединститутов к преподаванию курса «Основы информатики и ВТ»: Инструктивное письмо от 28 июня 1985 г. № 39—50 // Бюл. Министерства высшего и среднего спец. образования СССР. 1985. № 9. С. 9—10.

16. О подготовке студентов-выпускников государственных университетов к преподаванию в школе курса «Основы информатики и ВТ»: Приказ от 1 апреля 1985 г. № 243 // Бюл. Министерства высшего и среднего спец. образования СССР. 1985. № 6. С. 30—32.

17. О порядке организации и проведении подготовки учителей математики и физики средних общеобразовательных школ, преподавателей средних профессионально-технических училищ и средних специальных учебных заведений по курсу «Основы информатики и ВТ»: Приказ от 25 апр. 1985 г. № 74/303/65 // Бюл. Министерства высшего и среднего спец. образования СССР. 1985. № 8. С. 28—29.

18. О технической готовности школ к преподаванию Основ информатики // Учит. газета. 1986. 12 авг.

19. *Разумовский В. Г.* Реформа в действии: ЭВМ входит в жизнь школы // Физика в школе. 1985. № 4. С. 3—8.

20. *Хорошилов В., Красс Э.* Профессиональное образование и компьютерный всеобуч // Информатика и образование. 1986. № 1. С. 38—43.

О введении в системе профтехобразования нового общеобразовательного курса «Основы информатики и вычислительной техники», общетехнических предметов «Автоматизация производства на основе электронно-вычислительной техники» и «Основы программирования и ВТ». Обсуждаются вопросы, заданные представителями ряда ГПТУ страны.

21. *Шаповаленко С. Г.* Об организации в АПН СССР НИОКР по изучению и использованию в средних общеобразовательных школах и ПТУ электронно-

вычислительной, микропроцессорной техники и программирования. М., 1983.

22. Компьютеры и образование: Указатель советской литературы / Сост. Е. А. Павлова, И. И. Серебренникова. М., 1985.

23. *Резников З. М.* Физика, автоматика, ЭВМ: Обзор литературы за 1974—1984 гг. // Физика в школе. 1985. № 2. С. 56—59.

Раздел II

1. *Авербух А.* Алгоритмы для работы с табличными величинами: Метод. практикум по курсу «Основы информатики и ВТ» // Учит. газета. 1986. 4 февр.

2. *Авербух А.* Упорядочение линейной таблицы: Метод. практикум по курсу «Основы информатики и ВТ» // Учит. газета. 1986. 6 февр.

3. Алгоритм и его свойства: Метод. практикум. Основы ЭВМ — всем // Учит. газета. 1985. 14 сент.

4. Алгоритм работы с величинами (вопросы и ответы): Метод. практикум. Основы ЭВМ — всем // Учит. газета. 1985. 24 сент.

5. *Антипов И. Н.* Основы информатики в техникумах и училищах // Среднее спец. образование. 1985. № 8. С. 30—34.

В статье представлены основные понятия курса «Основы информатики и ВТ»: информация и способы ее переработки, алгоритмы и средства их описания, программирование и ЭВМ.

6. *Антипов И. Н.* Программирование на Бэйсике // Математика в школе. 1985. № 4. С. 48—52.

В статье описываются основные элементы языка Бейсик и элементарные приемы программирования, которые составляют ядро многих вариантов языка, ориентированных на различные ЭВМ.

7. *Антипов И. Н., Буковцев О. А., Шамурин В. Л.* Обучение учащихся VII класса работе на МК МКШ-2 // Математика в школе. 1985. № 3. С. 33—36.

8. *Антипов И., Кобринский Я., Смекалин Д.* Знакомство с языком алгоритмов: Алгоритмический язык в школьном обучении // Учит. газета. 1985. 27 июля.

9. *Антипов И., Кузнецов А., Бешенков С.* Про алгоритмы работы с величинами: Метод. практикум // Учит. газета. 1985. 30 июля.

10. *Апатова Н. В.* Методика подготовки пользователей ЭВМ в автоматизированной обучающей системе: Автореф. дис. ... канд. пед. наук. М., 1984.

11. *Балк М. Б., Полухин А. А.* О некоторых особенностях решения уравнений с помощью микрокалькулятора // Математика в школе. 1983. № 5. С. 35—39.

Авторы знакомят с возможностями реализации метода простых итераций с помощью МК.

12. Вспомогательные алгоритмы: Метод. практикум к курсу «Основы информатики и ВТ» // Учит. газета. 1986. 7 янв.

13. *Галицкий М.* Общие правила алгоритмического языка: Метод. практикум к курсу «Основы информатики и ВТ» //Учит. газета. 1985. 26 сент.

14. *Галицкий М.* Понятие алгоритма: Метод. практикум // Учит. газета. 1985. 17 сент.

15. *Галицкий М.* Формальное исполнение алгоритма: Метод. практикум //Учит. газета. 1985. 21 сент.

16. *Галицкий М., Рубашевский С.* Алгоритмы приближенного вычисления площадей: Метод. практикум по курсу «Основы информатики и ВТ» // Учит. газета. 1986. 20 марта.

17. *Галицкий М., Рубашевский С.* Построение алгоритмов для решения задач из курса математики. Вычисление значений многочлена. Схема Горнера: Метод. практикум по курсу «Основы информатики и ВТ» // Учит. газета. 1986. 27 февр.

18. *Гиглавый А. В.* и др. Учи работать с ЭВМ. М., 1984.

19. *Горский В. Д., Подгорнова Г. А.* Подготовка операторов ЭВМ на УПК // Школа и производство. 1981. № 6. С. 29—31.

20. *Гребень Г.* Телеэкран об информатике // Учит. газета. 1985. 21 сент.

21. *Дыга П., Балашова Д.* Компьютер — в учебный процесс // Народное образование. 1985. № 6. С. 79—81.

Статья посвящена опыту преподавания курса «Программирование и вычислительная математика» в школе № 33 г. Ярославля.

22. *Ермакова В. М., Утлинский Е. В.* Лабораторный практикум «Алгоритмический язык» // Информатика и образование. 1986. № 2. С. 30—43.

Содержание статьи представляет собой руководство по использованию Е-практикума при изучении II части курса «Основы информатики и вычислительной техники». Предназначено для школьников, учителей и студентов, изучающих информатику.

23. *Ермакова В., Утлинский Е.* Методический практикум по курсу «Основы информатики и вычислительной теники» // Учит. газета. 1986. 30 авг.

24. *Ермакова Е., Утлинский Е.* Методический практикум для учителей // Учит. газета. 1986. 6 сент.

25. *Ермакова В. М., Утлинский Е. В.* Методические рекомендации по проведению уроков информатики в X классе по теме «Алгоритмы работы с литерными величинами // Учит. газета. 1987.

26. *Ермакова В. М., Утлинский Е. В.* Методические рекомендации по проведению уроков информатики в X классе по теме «Язык программирования Рапира» //Учит. газета. 1987.

27. *Ермакова В. М., Утлинский Е. В.* Методические рекомендации по проведению уроков информатики в X классе по теме «Организация диалога на персональном компьютере» // Учит. газета. 1987.

28. *Ершов А. П.* Алгоритмический язык в школьном курсе информатики и ВТ // Микропроцессорные средства и системы. 1985. № 2. С. 48—57.

29. *Ершов А. А.* Компьютер — алгоритм — алгоритмический язык // Квант. 1985. № 9. С. 50—54.

30. *Ершов А. П.* Что такое информатика? Урок для учителей: Методический практикум для учителей // Учит. газета. 1985. 5 марта.

31. *Ершов А. П.* Алгоритмический язык: Для записи алгоритмов работы с величинами // Наука и жизнь. 1986. № 1. С. 138—141.

32. *Жегалин В. А., Первин Ю. А., Хоперсков А. Б.* Генератор задач — учебно-ориентировочный пакет прикладных программ // Прикладные методы математики. Новосибирск, 1980.

33. *Заварыкин В.* Будущему учителю — знания вычислительной техники: О подготовке кадров в педагогических институтах Российской Федерации // Народное образование. 1985. № 6. С. 32—35.

34. Зачем мы изучаем информатику?: Метод. практикум //Учит. газета. 1985. 12 сент.

35. *Звавич Д.* Исполнение алгоритма и таблица значений: Метод. практикум по курсу «Основы информатики и ВТ» // Учит. газета. 1985. 26 нояб.

36. Информатика за рубежом: Компьютеры и школы //Учит. газета. 1985. 15 авг.

37. *Карташова Р.* Алгоритмический язык: Итоговое занятие //Учит. газета. 1985. 17 окт.

38. *Карташова Р.* Команда ветвления: Метод. практикум к курсу «Основы информатики и ВТ» //Учит. газета. 1985. 10 окт.

39. *Король В.* ЭВМ ждет наставников: Проблемы применения ЭВМ в учебном процессе ПТУ //Профессионально-техническое образование. 1985. № 1. С. 26—28.

40. *Коуров В.* Этапы решения задач с использованием ЭВМ: Метод. практикум к курсу «Основы информатики и ВТ» //Учит. газета. 1986. 28 янв.

41. *Красс Э., Оксман В.* Информатика: шаг следующий //Профессионально-техническое образование. 1985. № 9. С. 10—12.

Авторы знакомят читателей со специфическими особенностями изучения нового курса в средних профтехучилищах.

42. *Кривоборский В. Е.* Обучение старшеклассников работе на электронно-вычислительной технике // Школа и производство. 1985. № 7. С. 33.

Директор УПК Ленинграда рассказывает в статье об опыте подготовки школьников по двум специальностям: оператор ЕС ЭВМ и оператор мини-ЭВМ.

43. *Кузнецов А. А., Нурмухамедов Г. М.* Компьютер на школьной парте // Физика в школе. 1985. № 4. С. 49—54.

В статье дается описание современных микро-ЭВМ, которыми оснащаются школы нашей страны, а также приводится описание комплекта учебной ВТ.

44. *Кузнецов А. А., Смекалин Д. О.* Как дать понятие об алгоритме?: Метод. практикум //Учит. газета. 1985. 23 июля.

45. *Кузнецов А. А., Нармухамедов Г. М.* Первоначальные представления об ЭВМ // Физика в школе. 1986. № 2. С. 40—44.

В статье рассказывается о разных поколениях ЭВМ, об устройстве и принципах действия ЭВМ.

46. Методические указания по преподаванию курса «Основы информатики и вычислительной техники» в X классе // Информатика и образование. 1986. № 1. С. 54—63; № 2. С. 44—63; № 3. С. 31—60.

Вниманию читателей предлагаются главы методического пособия по преподаванию курса «Основы информатики и вычислительной техники в X классе», которое готовится к выпуску в издательстве «Педагогика».

47. *Молочников А. Д.* Повышать компьютерную грамотность учащихся // Среднее специальное образование. 1985. № 7. С. 13—14.

В публикации отмечается, что на первых этапах решения проблемы компьютеризации образования важную роль могут сыграть МК «Электроника БЗ-34» и «Электроника М/56». Описывается методика их использования.

48. *Монахов В. М.* Новый предмет «Основы информатики и ВТ» // Народное образование. 1985. № 4. С. 84—86.

В статье раскрываются проблемы вооружения учащихся знаниями и навыками владения ЭВМ. Описываются цель и содержание курса.

49. *Монахов В. М., Бешенков С. А.* Этапы решения задач с помощью ЭВМ: Метод. практикум // Учит. газета. 1985. 3 авг.

50. *Монахов В. М., Кузнецов А. А., Смекалин Д. О.* Микропроцессорная техника в зарубежной школе // Сов. педагогика. 1984. № 8. С. 117—121.

Авторы знакомят читателей с опытом изучения информатики и программирования в зарубежной школе, а также с вопросами использования компьютера как средства обучения.

51. *Монахов В. М., Кузнецов А. А., Шварцбурд С. И.* Обеспечить компьютерную грамотность школьников // Сов. педагогика. 1985. № 1. С. 21—29.

52. *Орешков И. С.* Микропроцессоры и микро-ЭВМ — в учебный процесс: Рязанский техникум электронных приборов // Среднее спец. образование. 1983. № 12. С. 34—37.

Директор техникума делится опытом организации работы по подготовке специалистов по ВТ и программированию.

53. Основы ЭВМ — всем: Метод. практикум для учителей //Учит. газета. 1985. 20 авг.

В статье содержатся сведения о помощи вузов школам по созданию новой ЭВТ и ППП.

54. Основы ЭВМ — всем: Опыт школ г. Барнаула в усвоении и освоении компьютеров // Учит. газета. 1985. 19 марта.

55. Основы ЭВМ — всем: «Школьница» придет в школу // Учит. газета. 1985. 11 июля.

В статье публикуется беседа с автором системы «Школьница» — Ю. А. Первиным.

56. *Ратинский М.* Небезобидный крен // Учит. газета. 1986. 3 июня.

По мнению автора, рекомендованная для школы программа по «Основам информатики и ВТ» практически не отвечает задачам, стоящим перед новым учебным предметом.

57. *Соколов В.* Понятие вспомогательного алгоритма: Метод. практикум по курсу «Основы информатики и ВТ» //Учит. газета. 1986. 14 янв.

58. *Соколов В.* Последовательное построение алгоритма: Метод. практикум по курсу «Основы информатики и ВТ» // Учит. газета. 1986. 21 янв.

59. *Фидлер М.* Присваивание значения. Промежуточная величина: Метод. практикум по курсу «Основы информатики и ВТ» // Учит. газета. 1985. 19 ноября.

60. *Хорошева И., Гольц Я., Шень А.* Рекомендации по преподаванию курса «Основы информатики и ВТ» // Информатика и образование. 1986. № 1. С. 46—48.

61. *Цирюльникова А., Логвинов И.* Основы ЭВМ — всем: О компьютере и реальности // Учит. газета. 1986. 5 июля.

62. *Шаповаленко С. Г.* Об изучении и применении в средних школах электронно-вычислительной техники, микропроцессорной техники и программирования //Вопросы психологии. 1983. № 4. С. 11—15.

63. *Шень А.* Команды ветвления: Метод. практикум по курсу «Основы информатики и ВТ» // Учит. газета. 1985. 3 окт.

64. *Шоднев Д., Дик Ю. И.* Изучение физических основ устройства и действия ЭВМ // Физика в школе. 1986. № 2. С. 37—44.

65. *Яшин Г.* Компьютер — для всех // Учит. газета. 1986. 7 авг.

В статье говорится об опыте первого года обучения информатике в Свердловске.

Раздел III

1. *Барыкин К.* Компьютер по-болгарски: Опыт болгарской школы // Огонек. 1985. № 35.

2. *Бенедек А.* ЭВМ в венгерской школе // Народ. образование. 1985. № 6. С. 86—87.

3. *Болтянский В. Г., Григорян Э.* Микрокалькулятор в младших классах //Математика в школе. 1985. № 5. С. 24—29.

Авторы делятся опытом рационального использования МК. Они отмечают, что при правильном сочетании устного счета с работой на приборе учащиеся начальной школы (экспериментальные классы) быстрее овладевали навыками устных вычислений, глубже осознавали логические приемы решения задач, приобретали навыки математического мышления.

4. *Болтянский В. Г., Рубцов В. В.* Проблемы компьютеризации обучения // Математика в школе. 1986. № 1. С. 69—72.

Статья посвящена анализу материалов, обсуждавшихся на международной конференции «Дети в век информации: завтрашние проблемы — сегодня», проходившей в мае 1985 г. в НРБ.

5. *Бородкин Л. И.* ЭВМ в истории // Преподавание истории в школе. 1986. № 1. С. 9—14.

Автор рассказывает о предпосылках использования математических методов и ЭВМ в исторических исследованиях; об итогах и перспективах развития нового направления.

6. *Бутаев Б. С.* ЭВМ в химии // Химия в школе. 1985. № 4. С. 68—71.

Статья посвящена различным аспектам применения ЭВМ в современной химической науке, а также некоторым вопросам использования компьютеров в обучении химии.

7. *Бычков А. В.* Информатика во внеклассной и внешкольной работе // Физика в школе. 1984. № 6. С. 68—69.

В статье речь идет о различных формах ознакомления учащихся с информатикой, в частности об организации кружков в школах, при Дворцах пионеров, УПК и ВЦ. Приводится список научно-популярной литературы для школьников.

8. *Вальковская И.* ЭВМ проверяет контрольную ЭВМ в учебном процессе // Семья и школа. 1985. № 8. С. 8—9.

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

Авторы представляют некоторые требования к обучающим программам: к вводной части программ, содержанию текста, характеру реплик, комментариев.

10. *Городисский Ф.* ЭВМ и школьники: Об использовании современной ВТ в учебном процессе // Учит. газета. 1984. 19 мая.

11. *Звенигородский Г. А.* Система математического обеспечения, ориентированная на школьный учебный процесс // УСИМ. 1980. № 5. С. 76—82.

12. *Зелинская Г. Я.* Об оптимальной модели микрокалькулятора для школы // Математика в школе. 1984. № 3. С. 50—51.

В статье содержатся сведения о видах МК, их классификация, описываются требования к школьным МК.

13. *Дьячко А. Г.* Микропроцессоры в учебном процессе: С совещания экспертов ЮНЕСКО // Вестник высшей школы. 1985. № 12. С. 70—73.

14. *Извозчиков В. А., Марков И. В.* Школьники изучают ЭВМ // Физика в школе. 1984. № 4. С. 28—30.

Авторы знакомят читателей с различными методами изучения школьниками ЭВМ, важную роль отводят экскурсии на ВЦ.

15. *Ионов Г. Н.* Микрокалькулятор как средство обучения и контроля // Математика в школе. 1984. № 3. С. 51—54.

В статье приведены программы решения на МК наиболее распространенных типов школьных задач. Программы составлены на языке МК «Электроника БЗ-34». Они снабжены подробными инструкциями и комментариями.

16. *Ионов Г. Н.* Электронный помощник учителя: Программирование на МК на уроках математики // Математика в школе. 1983. № 5. С. 31—35.

17. *Кобринский Я. Н., Кузнецов А. А.* Особенности пакетов прикладных программ // Информатика и образование. 1986. № 3. С. 18—25.

Раскрываются основные предпосылки, определяющие назначение и состав пакетов прикладных программ, возможности, предоставляемые их использованием в учебном процессе. Описываются программы системного и функционального наполнения.

18. Компьютер — алгоритм творчества: О проблемах, возникающих при изучении основ информатики в профтехучилищах // Профессионально-техническое образование. 1986. № 8.

19. *Кошелева Е. А., Шилов В. Ф.* Об использовании микрокалькуляторов // Химия в школе. 1985. № 2. С. 41—45.

В статье рассказывается об эффективном опыте использования МК при решении расчетных химических задач. Приведены программы для МК.

20. *Кузнецов А. А., Глазунова М. А., Игошев Б. М.* Овладение компьютерной грамотностью в процессе технического творчества (школьников) // Советская педагогика. 1986. № 2. С. 29—32.

В статье обсуждается роль новых направлений технического творчества, возникающих с внедрением в обучение компьютерной техники, описывается ведущая роль внеклассной работы по новой технике — технический кружок.

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

22. *Кузнецов Е. Ю., Минкин Л. К.* Виды микрокалькуляторов // Математика в школе. 1982. № 4. С. 35—37.

В статье описаны педагогические и технические возможности МК, а также их типы.

23. *Лившиц М.* Серьезные игры на досуге // Учит. газета. 1985. 6 авг.

Автор рассматривает возможности применения ЭВМ во внеклассной работе, в частности — компьютерные игры, включающие элементы приключенческого характера.

24. *Литвинова А.* ЭВМ в школах Франции // Народное образование. 1985. № 7. С. 84—86.

Автор подробно описывает пути решения компьютеризации образования во Франции. Знакомит читателей с игровыми программами.

25. *Литерат С.* ЭВМ в учебном процессе: 130-я средняя школа г. Новосибирска // Народное образование. 1985. № 2. С. 48.

Публикация знакомит с опытом использования ЭВМ в экспериментальной школе г. Новосибирска. Перечисляются наиболее перспективные направления использования ВТ в учебном процессе.

26. *Малкова Т. В., Монахов В. М.* Математическое моделирование — необходимый компонент современной подготовки школьника // Математика в школе. 1984. № 3. С. 46—50.

27. Методические рекомендации по проектированию обучающих программ. Киев, 1986.

28. Мини-ЭВМ. Олимпиада: Об опыте приобщения учащейся молодежи Народной Республики Болгарии к компьютерной технике // Учит. газета. 1985. 4 апр.

29. *Низамов И. М.* Математическое моделирование физических явлений с помощью ЭВМ // Физика в школе. 1986. № 3. С. 59—64.

30. *Одияк Б. П.* Применение микро-ЭВМ при изучении физики в средней школе Франции // Физика в школе. 1984. № 6. С. 78—81.

В статье излагаются основные тенденции совершенствования методов использования ЭВМ при изучении физики в школах Франции, приводятся примеры обучающих программ.

31. *Оксман В.* Микрокалькуляторы: типовая тема на уроках математики ПТУ // Профессионально-техническое образование. 1983. № 11. С. 26—27.

В статье приведено примерное распределение учебного материала по урокам, описана методика изложения отдельных тем.

32. Персональный компьютер: перспективы близкие и далекие. Академик Велихов, академик Ершов и др. о феномене персонального компьютера // Наука и жизнь. 1985. № 10.

33. *Полат Е. С.* Компьютер и школа: Использование вычислительной техники в учебном процессе // Физика в школе. 1985. № 2. С. 51—55.

В публикации выделены основные подходы к внедрению компьютерного обучения в практику работы школы, СПТУ, вуза. Приводятся наиболее распространенные в мировой практике обучения виды программ для ЭВМ.

34. *Разумовская Н. В.* Компьютер на уроках физики // Физика в школе. 1985. № 3. С. 51—56.

В статье большое внимание уделяется программам моделирования на ЭВМ различных физических процессов.

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

Описываются возможности педагогических программных средств, их классификация, приводится комплекс требований к ППС.

36. *Савельев А. Я.* Проблемы автоматизации обучения // Вопросы психологии. 1986. № 2. С. 11—21.

37. *Савина А.* Компьютер в болгарской школе // Народное образование. 1985. № 8. С. 84—85.

Автор рассказывает об основных направлениях компьютеризации народного образования Болгарии, обсуждает психолого-педагогические аспекты этой проблемы.

38. *Савинов Ю. А., Щербина Ю. Д.* ЭВМ в американской школе: Тенденции и проблемы // США. Экономика. Политика. Идеология. 1985. № 11. С. 89—96.

39. *Сергеев Ф., Фридман Г.* Поговорим с тобой, компьютер // Учит. газета. 1985. 29 авг.

Авторы делятся опытом обучения школьников работе на ЭВМ в г. Омске.

40. *Сергеева Т. А.* Использование персональных ЭВМ в обучении химии // Химия в школе. 1985. № 5. С. 64—67.

В статье перечислены методические проблемы, связанные с внедрением ЭВМ в школу. Автор раскрывает основные аспекты использования компьютера в обучении химии.

41. *Сергеева Т. А.* Об обучающих программах для ЭВМ // Химия в школе. 1986. № 6. С. 45—49.

Освещаются два аспекта проблемы использования ЭВМ на уроках химии. Приводится примерный перечень обучающих программ по всему школьному курсу химии.

42. *Сердюков П. И.* ЭВМ в обучении иностранным языкам: Взгляд в будущее // Иностранные языки в школе. 1985. № 5. С. 6—9.

ЭВМ рассматривается авторами как средство обучения, используемое для эффективной организации самостоятельной работы учащихся.

43. *Старчинский Г.* Урок географии: Использование ЭВМ в учебном процессе // Техника — молодежи. 1985. № 7. С. 47.

В публикации в качестве примера представлена обучающая программа по географии.

44. *Труш В., Голод В.* Компьютеры в дефектологии // Информатика и образование. 1986. № 2. С. 24—28.

45. *Хоперсков А. Е.* Генератор задач по школьному курсу механики (программная реализация) // Оперативно-информационные материалы ВЦ СО АН СССР. Новосибирск, 1981.

46. ЭВМ в учебно-воспитательных учреждениях ГДР // Советская педагогика. 1986. № 2 с. 110—114.

Раздел IV

1. *Алексеев Н. Г., Юдин Б. Г.* Эволюция проблематики искусственного интеллекта // Вопросы философии. 1979. № 4. С. 91—95.

2. *Брусенцова Т. А.* О психолого-педагогических принципах компьютерной системы обучения ЛОГО // Вопросы психологии. 1986. № 2. С. 164—170.

3. *Габай Т. В., Морозова И. Г.* Психолого-педагогические требования к техническим средствам обучения для высших учебных заведений. М., 1984.

4. *Звягинцев В. А.* Проблема отношений человека и машины в компьютерной революции // Вопросы философии. 1986. № 3. С. 34—58.

Среди важных психолого-педагогических проблем внедрения ЭВМ в учебный процесс авторы называют проблемы создания обучающих программ, психического развития школьников и усиления индивидуализации обучения.

5. *Зуев К. А.* Социальные последствия компьютеризации в условиях капитализма // Вопросы философии. 1984. № 10. С. 79—86.

6. Искусственный интеллект и психология / Под ред. О. К. Тихомирова. М., 1976.

7. Компьютеризованная система проведения научных исследований, психодиагностики и обучения // Психологический журнал. 1985. Т. 6. № 6. С. 142—146.

8. *Корнилова Т. В., Белавина И. Т.* Использование «советов» ЭВМ при решении мыслительных задач // Вестник Моск. ун-та. Сер. Психология. 1978. № 3. С. 52—62.

В публикации отмечается, что решение мыслительной задачи, опосредованное диалоговыми общениями с ЭВМ, более эффективно, чем обычное безмашинное решение.

9. *Кочетков Г. Б.* Компьютерные игры: свет и тени // Микропроцессорные средства и системы. 1985. № 3. С. 16—20.

10. *Леонтьев А. Н.* Автоматизация и человек // Психологические исследования. М., 1970. Вып. 2.

В статье описываются два подхода к психологическому аспекту проблемы «автоматизация и человек».

11. *Ломов Б. Ф.* ЭВМ и развитие человека: Проблемы педагогической психологии // Вестник высшей школы. 1985. № 12. С. 29—33.

12. *Лысенко Е. Е.* Компьютерная игра с точки зрения психолога // Микропроцессорные средства и системы. 1985. № 3. С. 15—16.

В статье рассматриваются психологические требования к игре как виду человеческой деятельности, показывается роль игры в развитии детей, обозначаются пути решения проблемы использования компьютерных игр в обучении.

13. *Ляудис В. Я., Тихомиров О. К.* Психология и практика автоматизированного обучения // Вопросы психологии. 1983. № 6. С. 16—27.

Статья посвящена актуальным проблемам использования АОС, излагаются принципы организации коллективов разработчиков и пользователей АОС на базе ЭВМ.

14. *Машбиц Е. И.* Психологические проблемы обучения пользователей ЭВМ. Киев, 1976.

Основное внимание автор уделяет проблеме построения содержания обучения непрофессиональных пользователей ЭВМ.

15. *Машбиц Е. И.* Психолого-педагогические аспекты компьютеризации // Вестник высшей школы. 1986. № 4. С. 22—29.

16. *Метешкин А. А.* Чтобы преодолеть психологический барьер: ЭВМ в высшей школе // Вестник высшей школы. 1985. № 3. С. 32—33.

17. *Моисеев Н. Н., Фролов И. Т.* Высокое соприкосновение, общество, человек и природа в век микроэлектроники, информатики и биотехнологии // Вопросы философии. 1984. № 9. С. 24—41.

18. *Монахов В. М.* Психолого-педагогические проблемы обеспечения компьютерной грамотности // Вопросы психологии. 1985. № 3. С. 14—23.

Статья посвящена анализу методологической и методической готовности педагогической науки к введению нового курса «Основы информации и ВТ».

19. *Попов Э. В.* Системы общения человека с ЭВМ на естественном языке // Вопросы философии. 1979. № 4. С. 76—78.

За «круглым столом» журнала обсуждаются социально-философские проблемы человеко-машинных систем.

20. *Смоляк Г. Л.* Человек и компьютер: Социально-философские аспекты автоматизации управления и обработки информации. М., 1981.

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

22. *Тихомиров О. К., Бабанин Л. А.* ЭВМ и новые проблемы психологии. М., 1986.

23. *Тихомиров О. К.* Психолог и робот // Знание — сила. 1980. № 8. С. 18—20.

В публикации с оригинальных позиций рассматривается вопрос: «Что дает науке кибернетический бум, в чем его психологические мотивы?»

24. *Тихомиров О. К.* Психологическая структура диалога «Человек — ЭВМ» // Вестник МГУ. Сер. Психология. 1984. № 2. С. 17—24.

Автор утверждает, что диалог человека с компьютером характеризуется не только логико-лингвистической структурой, но и психологической. В статье раскрываются компоненты психологической структуры такого диалога.

25. *Тихомиров О. К., Белавина И. Г.* Интеллектуальная деятельность в условиях «диалога» с ЭВМ // Интеллект человека и программы ЭВМ. М., 1979. С. 11—14.

26. *Тихомиров О. К., Белавина И. Г., Войскунский А. Е.* Психология и практика программного обеспечения ЭВМ // Вестник МГУ. Сер. Психология. 1981. № 1. С. 3—14.

Авторы обсуждают ряд психологических проблем, связанных с разработкой и использованием программного обеспечения.

27. *Тихомиров О. К., Войскунский А. Е.* Временны́е характеристики решения задач в условиях диалога человека с ЭВМ // Вопросы психологии. 1980. № 6. С. 155—160.

28. Человек и компьютер / Под ред. О. К. Тихомирова. М., 1972. Вып. 1.

В сборнике рассматриваются психологические вопросы взаимодействия человека с ЭВМ: природа этого взаимодействия, структура, способы и средства коммуникации человека и ЭВМ.

29. *Шеншев Л. В.* Компьютеризация школьного образования как комплексная проблема // Вопросы психологии. 1987. № 1. С. 89—98.

Журнал в журнале
для школьников, студентов,
учащихся СПТУ
и техникумов

Издается при участии
ЦК ВЛКСМ

МОЛОДЕЖНАЯ ИНИЦИАТИВА

3

В НОМЕРЕ:

Цветная печать на «Агате»

Имея комплект «Агата» с принтером, трудно удержаться от соблазна печатать на нем рисунки. Система команд принтеров СРА-80, D-100 и FPSON FX-85, которые поставляются обычно с «Агатом», позволяет использовать так называемый растровый режим (*bit image mode*), в котором поступающий в буфер байт несет информацию о состоянии всех проволочек печатающей головки. Конечно, хочется использовать этот режим, но:

стандартный драйвер CPAD100 не пропускает 7-й (самый старший) бит, который необходим для передачи как графической, так и некоторой управляющей информации;

удобнее, естественно, распечатывать рисунки, созданные в видеопамяти «Агата», а не задавать отдельные массивы данных.

Эти трудности решает программа ALVPR, преобразующая в данные для принтера черно-белую картинку второй страницы в высшем графическом формате и управляющая ее выводом на принтер.

На этом можно было бы и успокоиться, но возникло желание печатать и цветные картинки. Естественно, передать цвета можно лишь условно, кодируя полихромные точки совокупностью монохромных (цвета ленты). Предлагаемый алгоритм преобразует 8К байт цветного изображения среднего графического режима в 8К байт высшей графики той же страницы, кодируя каждый цветной пиксел соответствующим черно-белым квадратом 2·2.

Для начала вспомним, что одна цветная точка несет в себе 4 бита информации, а монохромная — только 1 (таким образом, для ГСР и для ГВР нужно 4· (128· 128)/ (8· 1024)=1·(256· 256)/ (8· 1024)=8К байт ОЗУ). Задача сводится к следующему: взять одну из тетрад цветного изображения, разбить ее на две части по 2 бита и перенести их «друг под другом» в соответствующие байты черно-белого формата. Так как области памяти ГСР-2 и ГВР-2 совпадают, то необходим небольшой буфер длиной в одну цветную строку (адреса $6000—$603F). Вот как выглядит преобразование байта с адресом $4000, если условно обозначить каждый бит цифрой:

```
                 -+---------+-
$4000:             !7654 3210!
                 -+---------+-

                 -+---------+-
$4000:             !76 32 ** **!
                 -+---------+-
$4020:             !54 10 ** **!
                 -+---------+-
```

Таким образом, из двух точек (7654) и (3210) формата ГСР получилось восемь точек в ГВР, условно распределенных по квадратам 2 · 2.

Детализировать алгоритм вряд ли стоит: любознательный читатель может самостоятельно в нем разобраться по предлагаемой распечатке дизассемблера программы USLPR.

Как ее использовать? Поместив во вторую графическую страницу полихромное изображение (созданное, например, программой GREDIT, автор — В. Г. Яблонский), включите принтер, вставьте лист бумаги и запустите USLPR на счет. Если же вам нужно только преобразование, отделите ALVPR, входящую в ее состав, командой RTS:

61B2—60 RTS

Дизассемблер текста USLPR

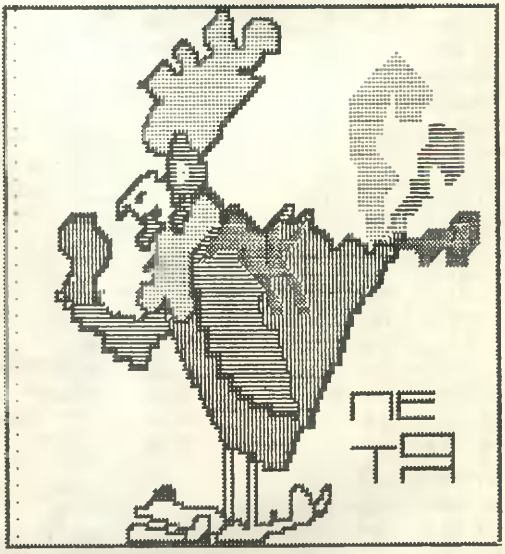
```
*** Инициализация: ***

6100-  A9 01     LDA  #$01
6102-  85 0F     STA  $0F
6104-  A9 FF     LDA  #$FF
6106-  85 09     STA  $09
6108-  85 0B     STA  $0B
610A-  A9 3F     LDA  #$3F
610C-  85 0C     STA  $0C
610E-  85 0A     STA  $0A
6110-  A9 FF     LDA  #$FF
6112-  85 07     STA  $07
6114-  A9 5F     LDA  #$5F
6116-  85 08     STA  $08
6118-  EA        NOP

*** Непосредственно текст: ***

6119-  E6 0B     INC  $0B
611B-  D0 02     BNE  $611F
611D-  E6 0C     INC  $0C
611F-  A0 00     LDY  #$00
6121-  B1 0B     LDA  ($0B),Y
6123-  85 00     STA  $00
6125-  29 C0     AND  #$C0
6127-  85 01     STA  $01
6129-  A5 00     LDA  $00
612B-  29 0C     AND  #$0C
612D-  2A        ROL
612E-  2A        ROL
612F-  05 01     ORA  $01
6131-  85 03     STA  $03
6133-  A5 00     LDA  $00
6135-  29 30     AND  #$30
6137-  6A        ROR
6138-  6A        ROR

6139-  85 02     STA  $02
613B-  A5 00     LDA  $00
613D-  29 03     AND  #$03
613F-  05 02     ORA  $02
6141-  85 04     STA  $04
6143-  A5 0F     LDA  $0F
6145-  F0 14     BEQ  $615B
6147-  A5 03     LDA  $03
6149-  85 05     STA  $05
614B-  A5 04     LDA  $04
614D-  2A        ROL
614E-  2A        ROL
614F-  2A        ROL
6150-  2A        ROL
6151-  85 06     STA  $06
6153-  A9 00     LDA  #$00
6155-  85 0F     STA  $0F
6157-  4C 19 61  JMP  $6119
615A-  EA        NOP
615B-  E6 07     INC  $07
615D-  D0 02     BNE  $6161
615F-  E6 08     INC  $08
6161-  A5 03     LDA  $03
6163-  6A        ROR
6164-  6A        ROR
6165-  6A        ROR
6166-  6A        ROR
6167-  05 05     ORA  $05
6169-  A0 00     LDY  #$00
616B-  91 07     STA  ($07),Y
616D-  A5 04     LDA  $04
616F-  05 06     ORA  $06
6171-  A0 20     LDY  #$20
6173-  91 07     STA  ($07),Y
6175-  A9 01     LDA  #$01
6177-  85 0F     STA  $0F
6179-  A5 07     LDA  $07
617B-  C9 1F     CMP  #$1F
617D-  D0 9A     BNE  $6119
617F-  EA        NOP
6180-  A9 FF     LDA  #$FF
6182-  85 07     STA  $07
6184-  A9 5F     LDA  #$5F
6186-  85 08     STA  $08
6188-  A0 41     LDY  #$41
618A-  88        DEY
618B-  F0 08     BEQ  $6195
618D-  B1 07     LDA  ($07),Y
618F-  91 09     STA  ($09),Y
6191-  4C 8A 61  JMP  $618A
6194-  EA        NOP
6195-  A5 0A     LDA  $0A
6197-  C9 5F     CMP  #$5F
6199-  D0 07     BNE  $61A2
619B-  A5 09     LDA  $09
619D-  C9 BF     CMP  #$BF
619F-  F0 10     BEQ  $61B1
61A1-  EA        NOP
61A2-  18        CLC
61A3-  A5 09     LDA  $09
61A5-  69 40     ADC  #$40
61A7-  85 09     STA  $09
61A9-  90 02     BCC  $61AD
61AB-  E6 0A     INC  $0A
61AD-  4C 19 61  JMP  $6119
61B0-  EA        NOP
61B1-  EA        NOP
61B2-  EA        NOP
```

Дизассемблер текста ALVPR
(автор — А. Голов, ЛЭМЗ)

```
61B3-  A2 03      LDX  #$03
61B5-  BD 29 61   LDA  $6129,X
61B8-  20 0A 61   JSR  $610A
61BB-  CA         DEX
61BC-  10 F7      BPL  $61B5
61BE-  A9 40      LDA  #$40
61C0-  85 01      STA  $01
61C2-  A9 20      LDA  #$20
61C4-  85 02      STA  $02
61C6-  A0 00      LDY  #$00
61C8-  84 00      STY  $00
61CA-  A2 05      LDX  #$05
61CC-  BD 2D 62   LDA  $622D,X
61CF-  20 0A 62   JSR  $620A
61D2-  CA         DEX
61D3-  10 F7      BPL  $61CC
61D5-  A9 00      LDA  #$00
61D7-  85 03      STA  $03
61D9-  A0 00      LDY  #$00
61DB-  84 04      STY  $04
61DD-  A6 03      LDX  $03
61DF-  B1 00      LDA  ($00),Y
61E1-  0A         ASL
61E2-  CA         DEX
61E3-  10 FC      BPL  $61E1
61E5-  26 04      ROL  $04
61E7-  18         CLC
61E8-  98         TYA
61E9-  69 20      ADC  #$20
61EB-  A8         TAY
61EC-  D0 EF      BNE  $61DD
61EE-  A5 04      LDA  $04
61F0-  20 0A 62   JSR  $620A
61F3-  E6 03      INC  $03
61F5-  A5 03      LDA  $03
61F7-  C9 08      CMP  #$08
61F9-  90 DE      BCC  $61D9
61FB-  E6 00      INC  $00
61FD-  A5 00      LDA  $00
61FF-  C9 20      CMP  #$20
6201-  90 D2      BCC  $61D5
6203-  E6 01      INC  $01
6205-  C6 02      DEC  $02
6207-  D0 BD      BNE  $61C6
6209-  60         RTS
620A-  8D D0 C0   STA  $C0D0
620D-  CD D0 C0   CMP  $C0D0
6210-  F0 08      BEQ  $621A
6212-  A9 89      LDA  #$89
6214-  8D D3 C0   STA  $C0D3
6217-  20 20 62   JSR  $6220
621A-  AD D2 C0   LDA  $C0D2
621D-  30 FB      BMI  $621A
621F-  0A         ASL
6220-  8D D1 C0   STA  $C0D1
6223-  A9 C0      LDA  #$C0
6225-  8D D1 C0   STA  $C0D1
6228-  60         RTS
```

Для программы ALVPR необходимы данные:
6229— 08 41 1B 10 01 00 4B 1B 09 0A F7

Образец работы алгоритма



Е. ГОПЕНГАУЗ
г. Электросталь, средняя школа № 20, X класс

Примечание. В распечатку вкралась ошибка: во второй и третьей строках программы ALVPR вместо 61 должно стоять 62.

Письмо из Парижа

Дорогой друг!

Давно не писала, потому что события у меня происходили невероятные. Ты, конечно, знаешь моего юного друга Ансельма Лантюрлю, любознательного и трудолюбивого, хотя и наивного изобретателя. Пока он мастерил паровые копии моего ученого пеликана Леона и гениальной улитки Терезии, все было ничего, хотя он и перевернул всю гостиную вверх дном.

Началось все, когда Ансельм решил научить умуразуму пылесос. Сначала он сделал «программируемый автомат» (т. е. пылесос-черепаху), от которого сразу сбежал мой кот. Проанализировав ситуацию, Ансельм сделал «систему с входом и выходом», в результате чего чуть не погибла Терезия. Тогда Ансельму пришлось заняться «анализом сигнала», и все было бы хорошо, но Ансельм захотел сделать пылесос-черепаху максимально быстрой и сразу столкнулся с «временем реакции», а введя «обратную связь», познакомился с понятием «порядка системы с обратной связью».

И если у Ансельма неистощимая энергия, то пылесос должен был зарядить свои аккумуляторы. Так появились «саморегулирующиеся системы», и мы поняли, что вся созданная человеком техника и технология — это способы воспроизведения или дополнения природных процессов.

Дальше я уже не вытерпела, и мы вместе занялись «распознаванием образов», познакомились с «шумом», «обработкой зрительной информации», «восприятием», «разумом и глупостью».

Мой научный руководитель Жан-Пьер Пети, которого ты знаешь как специалиста по физике плазмы, астрофизике и кибернетике, узнав про наши эксперименты, нарисовал всех нас, проявив при этом прекрасный юмор и талант художника-графика. Кроме того, он заметил, что мы пошли по стопам Норберта Винера, и вместе с нами нафантазировал про «искусственный разум» и «более или менее отдаленное будущее».

И чтобы ты смог убедиться, как увлекательны были наши эксперименты, профессора В. С. Голубев и Б. А. Рябов перевели на русский язык те немногочисленные

диалоги, которые уместились на свободных от рисунков местах, а издательство «Мир», прекрасно оформив, издало эти рисунки отдельной книгой под названием «О чем размышляют роботы?».

Конечно, ты можешь подумать, что все нарисованное в этой книге только шутка, и ошибешься. Попробуй после того, как посмеешься над нашими неудачами, перечитать «Человека-машину» Ламетри или «Кибернетику» Винера. Даже «Теория автоматического регулирования» на IV курсе вуза тебе покажется знакомой.

На этом прощаюсь.

Твоя подруга Софи.

Письмо доставил
А. Архангельский.

Яркий снимок... Но он вызывает разные ассоциации у разных людей.

Физиолог скорее всего поинтересуется: а знают ли те, кто занимается с этими детьми, о строжайших ограничениях по времени. Такие ограничения существуют при занятиях на ЭВМ с детьми любого возраста, особенно с малышами.

Математик, возможно, порадуется.

Фотограф скажет свое слово.

А что думаете вы, ребята?

Мы знаем, что об информатике у вас тоже нет единого мнения. Конечно, наш журнал не станет читать тот, кто совершенно безразличен к ней или даже не любит этот предмет. Но все же хотелось бы прочитать в ваших письмах, как вам дается этот курс, чем может помочь журнал, и в частности «Молодежная инициатива», в изучении информатики.

Ждем писем.

Ответы на кроссворд

По горизонтали: 5. «Максима». 7. Принтер. 8. Цифра. 9. Круиз. 10. Константа. 14. Поиск. 17. Сцена. 18. Емкость. 19. Маска. 20. Алгол. 22. Софтвер. 23. Гевея. 24. Филип. 29. Обработка. 31. Фраза. 32. Метка. 33. Автомат. 34. Нормаль.

По вертикали: 1. Варикап. 2. «Искра». 3. Эндрю. 4. Медиана. 6. Аргон. 7. Плата. 11. Триод. 12. Ассемблер. 13. Сцепление. 15. Омикрон. 16. Столбец. 21. Строб. 23. Габрово. 25. Паскаль. 26. Мбайт. 27. Экран. 28. Взвод. 30. Ферма.

Компьютерная Неделя в Красноярске

В школе № 41 г. Красноярска семь апрельских дней были посвящены празднику «Компьютерная неделя».

Не удивительно, что школа, расположенная в академгородке (ее шефами являются институты и ВЦ Красноярского филиала СО АН СССР), имеет базовый КУВТ «Ямаха».

Вызвавший всеобщий интерес кабинет работал весь фактически первый год своего существования с максимальной нагрузкой. Кроме учащихся школы № 41 в нем также проходили практику десятиклассники 8 школ района. Но утолить компьютерный голод даже тех счастливцев, которые имели возможность работать в кабинете по 4 часа в неделю, не удалось и в малой степени. Десятки ребят с надеждой и робостью заглядывали в класс на переменах. Как удовлетворить их любопытство и энтузиазм? В кружки при кабинете записалось около 500 школьников (при наборе 60).

В школе решили на одну неделю отменить занятия в кабинете, открыть его для всех желающих. Подготовка к компьютерной неделе (КН) началась за полтора месяца. Был создан оргкомитет, в который вошли старшеклассники, учителя, сотрудники ВЦ. Была разработана программа недели.

Конечно, не везде есть такие условия, как в школе № 41, но этот праздник для детей очень нужен. Вероятно, какие-то элементы сценария красноярской КН будут интересны и полезны при проведении подобных «компьютерных праздников» в других школах страны.

В первой половине дня по заранее составленному графику, учитывающему возможности программного обеспечения, школьники разных классов с I по X, работали в кабинете. Кто с помощью тренажеров изучал клавиатуру, кто смотрел демонстрационные и обучающие программы, решал задачи.

В 14 часов ежедневно ведущие ученые СО АН СССР читали лекции для школьников всех возрастов. Например: «Компьютеры в нашей жизни» (II—V классы); «Почему деревья не растут до неба» (VI—VII классы) и др.

Затем начиналась конкурсная программа. Ежедневно проводилось два конкурса: один — для младших, другой для старших школьников. Конкурсы для маленьких — «Мелодия», «Азбука», «Рисунок», «Английский язык» — не требовали ни знания машин, ни знания языков программирования. Во время их проведения дети знакомились с возможностями компьютера, изучали простейшие средства общения с ним.

Конкурс **«Азбука»**: после 15-минутного обучения работе на клавиатуре предлагалась 30-минутная контролирующая программа в виде следующего троеборья: русские слова, английские слова; числа. При оценке учитывалась правильность и скорость набора. Победители определялись по сумме полученных за каждый вид троеборья очков.

Конкурс **«Рисунок»**: с помощью специальной программы (похожей на известную игру «Волшебный экран») на экране монитора за 30 минут нужно было создать оригинальный рисунок, проявив максимум фантазии и находчивости.

Огромный интерес вызвал конкурс **«Английский язык»**. Это объясняется и английской специализацией школы, и прекрасным пакетом программ по английскому языку, разработанным сотрудницей НИВЦ (г. Пущино) Н. Л. Луниной.

Конкурс **«Спринт»** предназначался для тех, кто хорошо знал клавиатуру, умел быстро набирать текст. В медленном темпе зачитывался небольшой текст. Работа оценивалась по числу правильно набранных слов.

Три конкурса были посвящены работе с графикой «MSX — BASIC». Два из них («Графика» и «Эмблема»), о которых было объявлено заранее, предварительные. **«Графика»** — конкурс на самую интересную заставку к программе. Заставки были продемонстрированы разнообразные: рекламные надписи, картинки с элементами мультипликации, даже мультфильм. **«Эмблема»** — конкурс на заставку, общую для программ, выполненных в школе № 41. И хотя не удалось выбрать заставку-эталон, работы оказались очень яркими, непохожими друг на друга. Было решено продолжить конкурс вне рамок КН.

«Блиц» — конкурс-сюрприз. Используя графические средства языка «MSX — BASIC», за 45 минут нужно было нарисовать оригинальную картинку.

Три конкурса посвящались демонстрации музыкальных средств компьютера. **«Мелодия»** (для младших школьников): с помощью музыкального редактора заранее были записаны 10 популярных детских мелодий. Ребята должны были их узнать. **«Позывные»** (предварительный): нужно было написать позывные КН. Это задание потребовало от участников владения как компьютерной, так и музыкальной грамотой. Конкурс **«Аранжировка»** собрал много участников (так как задание не требовало знания машины). Предлагалось с помощью музыкального редактора аранжировать мелодию.

В первый же день КН был проведен **конкурс программистов,** в результате которого выяснилось, что его организация и правила проведения требуют доработки.

Не были забыты и взрослые. Но, к сожалению, игровой конкурс для учителей **«Пентамино»** собрал мало участников.

После окончания конкурсной программы каждый заранее записавшийся школьник мог прийти в кабинет и в течение двух часов заниматься тем, чем хотел: готовиться к конкурсам, к конференции, изучать языки программирования, работать с разного рода редакторами, решать задачи и т. д. Для компьютерных игр было отведено специальное время.

Увенчала КН **конференция по программированию**, на которой члены кружков отчитывались о своей работе за год. Было представлено около 30 докладов. Через час после окончания конференции и подведения итогов состоялось награждение и закрытие КН. Никто не остался без наград. А самой желанной была 4-часовая компьютерная игра.

Всем участникам конкурсной программы вручались значки. Каждый конкурс имел свой значок. Призеры получали разноцветные: красный — 1 место, желтый — 2, зеленый — 3. Главными призами были книги: «Компьютеры» (справочник в 3 томах); Пул Л. «Работа на персональном компьютере»; «Словарь по вычислительной технике» и др. Кроме того, в зависимости от профиля конкурса и возраста участников, призами были магнитофонные кассеты и грампластинки, канцелярские товары и игрушки.

Активными участниками конкурсной программы КН стали 200 школьников. А всего за неделю кабинет посетили около 600 учащихся, из них 70 % сели за компьютер впервые.

Когда на следующий день после окончания КН я пришел в школу, еще издали увидел у дверей кабинета стайку третьеклассников. Один, смело шагнув навстречу, спросил: «А когда можно записаться на компьютеры?» Для них КН продолжалась.

Е. КУЗНЕЦОВ

Двадцать программ студенческого отряда

Для эффективного внедрения ЭВМ в школу необходимо решить две проблемы.

Первая проблема — недостаток педагогических программных средств. Сейчас она усугубляется разнотипностью вычислительной техники, которой располагают школы.

Вторая проблема возникнет в ближайшие годы. Массовое внедрение ЭВТ в учебный процесс потребует от разработчиков программ обеспечить эффективность ее использования в народном образовании (Иначе государству будет выгоднее использовать эту технику в других отраслях народного хозяйства.) Эффективность же может быть обеспечена только созданием компьютерных технологий обучения, значительно отличающихся от традиционного обучения.

Как и всякая сложная информационная технология, компьютерная технология обучения требует коллективной деятельности пользователей. Во-первых, это обусловлено содержанием учебно-воспитательной деятельности, когда с данными о каком-то ученике работает несколько учителей, когда пакет программ, реализующий какой-либо метод обучения, используется в преподавании разных предметов и для обучения детей разного возраста.

Во-вторых, коллективная деятельность определяется сложностью компьютерных технологий, когда один человек физически не в состоянии владеть навыками работы со всеми компонентами технологии. Это приводит к специализации (системный программист, прикладной программист, инженер, ведущий задачу) и последующей кооперации.

Готов ли коллектив обычной школы внедрить и эксплуатировать компьютерную технологию обучения? Нет, не готов. Как по объективным, так и по субъективным причинам.

Напомним такой факт: обучение профессиональных программистов работе с новой операционной системой требует от двух до четырех месяцев с отрывом от производства.

Поскольку учителя не имеют опыта работы в информационных технологиях, то срок обучения необходимо увеличить вдвое.

К субъективным причинам, тормозящим внедрение компьютерных технологий обучения, относится явление, называемое «психологический барьер». Концептуальные модели, на базе которых проектируются технология, ее компоненты и функционирование, во многом определяют психологические схемы, которые необходимы пользователю. И кроме того, работа с технологией должна иметь для пользователя личностный смысл.

Обычно для снятия психологического барьера пользователя на длительное время включают в число разработчиков технологии. Т. е. из-за объективных и субъективных причин мы вынуждены будем для внедрения технологии обучения на год отвлечь весь учительский коллектив школы. Практически это вряд ли возможно.

Учитывая вышеуказанное, в системе народного образования должен быть создан организационный механизм, обеспечивающий эффективное внедрение и использование вычислительной техники в учебном процессе.

С нашей точки зрения, этот механизм должен работать следующим образом.

Из абитуриентов, сдавших экзамены в пединститут, создаются экспериментальные учебные группы по различным учительским специальностям. В рамках факультативов группы получают первоначальные знания о применении ЭВТ в учебном процессе и практические навыки работы с ней.

Из числа участников экспериментальных групп формируется постоянно действующий студенческий научно-производственный отряд (СНПО) по разработке педагогических программных средств (ППС). За время работы в СНПО студенты получают опыт разработки и применения ППС по своей учительской специальности, навыки организации и проведения летних школ юных программистов, учатся использовать ЭВМ для внеклассной работы.

Поскольку подготовка квалифицированного разработчика ППС занимает полтора-два года, оптимальным вариантом для существования СНПО должен быть достаточно длительный хозяйственный договор между ним и Управлением информатики Министерства просвещения — органом, определяющим долговременную политику в разработке ППС.

За год-полтора до окончания членами СНПО института из них должен быть организован временный молодежный творческий коллектив (ВМТК).

В момент создания ВМТК Управление информатики утверждает техническое задание на разработку ППС, заключает с ним хозяйственный договор, выделяет комплект учебной вычислительной техники.

Желательно решить вопрос о том, чтобы разработанные членами ВМТК педагогические программные средства могли бы служить формой сдачи госэкзамена или защитой дипломного проекта.

Одновременно с этим областной отдел образования определяет строящуюся школу, куда ВМТК по окончании своей деятельности совместно с техникой, разработанными ППС переходит в качестве школьного коллектива. Облоно определяет и включает в состав ВМТК учителей, которых в дальнейшем предполагает назначить руководителями школы.

Пединститут обеспечивает ВМТК производственными площадями и эксплуатационными материалами, осуществляет научно-методическое руководство разработками, ведет обучение членов ВМТК, совместно разрабатывает ППС.

Созданная таким образом школа довольно быстро может стать методическим центром по компьютеризации обучения. С ней желательно продолжить хозяйственный договор на разработку ППС. Это создало бы условия для вовлечения школьников в производительный труд, привело бы к соединению его с обучением информатике.

Такова теоретическая модель. Начиная с 1984 г. Новосибирский государственный педагогический институт практически работает в этом направлении. В 1984 и 1985 гг. экспериментальные группы были созданы из студентов - первокурсников физико - математического факультета, в 1986 — из первокурсников факультета начального обучения. В рамках факультатива эти группы изучали программирование, методику применения ЭВМ в учебном процессе. В мае 1986 г. приказом ректора при вычислительной лаборатории НГПИ был создан студенческий научно-производственный отряд по разработке педагогических программных средств. В отряд принимались студенты экспериментальных групп, добровольно изъявившие желание работать в летнее время в СНПО. В состав отряда вошло 14 студентов и 4 школьника, окончивших IX класс и в течение полугода до этого обучавшихся в вычислительной лаборатории. При наличии хоздоговора на разработку ППС для нас не составило бы труда увеличить численность отряда в 5—7 раз. Членами отряда создано 20 программ общим объемом около 110 Кбайт. Разработанные членом отряда О. Шамшурой, студентом III курса физико-математического факультета, программы вошли в состав пакета «Информатика-86», который сейчас тиражируется для всех школ.

Организованный как летний, отряд в настоящее время превратился в постоянно действующий. Это объективно обусловлено тем, что создание ППС — трудоемкий и длительный процесс и требует высокой квалификации разработчиков. По сути, параллельно с разработками идет индивидуальное обучение членов отряда уже в течение двух-трех лет.

Также следует отметить, что необходимым условием существования и продуктивной деятельности СНПО — разработчика ППС является наличие группы сотрудников института, экономически заинтересованных в разработке и внедрении компьютерных технологий обучения. Нашему институту в текущей пятилетке поручена разработка госбюджетной темы «Применение ЭВМ в учебном процессе вуза и школы». С нашей точки зрения, эффективность работы преподавателей, участвующих в теме, и, как следствие этого, деятельность СНПО могут быть значительно улучшены за счет изменения форм оплаты труда. Длительная, трудоемкая, коллективная работа по созданию ППС, с одной стороны, и ежемесячная зарплата без дифференциации за учебную и научную деятельность, с другой стороны, конечно же, не способствуют росту объемов и качества работ, не позволяют администрации института эффективно управлять научной деятельностью. Оптимальнее было бы, если бы для выполнения темы «Применение ЭВМ в учебном процесссе» из пятилетнего фонда заработной платы института была веделена определенная сумма, в пределах которой администрация могла бы заключать трудовые договоры с разработчиками при оплате труда по конечному результату. В дальнейшем эти экономические отношения могли бы быть перенесены на деятельность СНПО.

А. ЖАФЯРОВ,
А. СЕРЫХ

Приглашаем О. Шамшуру и его коллег из СНПО участвовать в работе журнала «Молодежная инициатива»!

Точка зрения

Информатика — фундаментальная естественная наука, изучающая процессы передачи и обработки информации.

А. П. Ершов

Раньше я думала, что компьютер — это интересно, а теперь поняла, что это такая же нудятина, как и многие другие школьные предметы.

Десятиклассница

Е. ЛИЛИТКО

Чего мы хотим?

Действительно, каких целей мы хотим достичь вводом в школьную программу нового курса? Уже настала пора разобраться, решают ли «Основы информатики и вычислительной техники» те задачи, которые перед ними ставились. Хотя цели и задачи нового школьного предмета были четко определены при его введении, многие преподаватели в своей практической деятельности придерживаются самых различных позиций, иногда очень далеких от первоначальных целей, и с этих позиций ведут преподавание. Более того: как ни парадоксально, но на это их в значительной мере толкают существующие учебная программа и пробное учебное пособие.

Приведем три наиболее характерные точки зрения. Новый курс введен для того, чтобы:

1) дать понятие о вычислительной технике и процессах передачи и обработки информации на уровне «что-то такое слышал»;

2) подготовить профессиональных программистов;

3) подготовить школьников к жизни в информационном обществе, т. е. обществе, ключевым ресурсом которого является информация и где использование компьютера не менее естественно, чем использование, например, автомобиля.

Не вызывает сомнения тот факт, что курс информатики в школе должен ставить перед собой именно третью задачу. Однако посмотрим, что мы имеем на сегодня.

Первая задача выполняется достаточно успешно. Особенно это касается безмашинного обучения, хотя и при наличии машин дела обстоят не намного лучше.

Вторая задача невыполнима. Ведь программист — это человек, знающий не только основы языка программирования и общую схему работы ЭВМ. Его знания простираются гораздо дальше, для их усвоения школьнику просто не хватит математической и общенаучной культуры. Кое-кто, возможно, не согласится с автором. В последнее время появилось очень много людей, которые знают несколько ключевых слов Бейсика и на этом основании считают себя программистами; печальное заблуждение! Автор слышал слова «пенальти» и «голкипер», однако футболистом себя не считает.

И наконец третья задача. Школьный курс информатики нисколько не способствует ее решению. В самом деле, чему мы учим детей? В программе выделено семь основных тем:

алгоритмы;
алгоритмический язык;
алгоритмы работы с величинами;
построение алгоритмов для решения задач;
принципы устройства и работа ЭВМ;
знакомство с программированием;
роль ЭВМ в современном обществе.

При взгляде на них создается впечатление, что если школьник научился строить алгоритмы типа «схема Горнера» и приблизительно знает, как работает процессор, то он овладел основами информатики. Но это, к сожалению, далеко не так. Найдется ли среди людей, по роду своей деятельности не связанных с математикой, хоть один, кому приходилось в жизни применять ту же схему Горнера или что-нибудь подобное? Нужны ли кассирам, работающим с системой «Сирена», знания алгоритмического языка? Слышал ли бухгалтер, рассчитывающий зарплату на ЭВМ, слово «Бейсик»?

В то время как развитие вычислительной техники во всем мире идет к тому, что тысячи людей будут использовать компьютер для своих целей без посредства программистов, мы ввели поголовное обучение

основам программирования (а никак не информатики).

Приведенное ниже не претендует на статус учебной программы или ее проекта. Это лишь соображения, из которых, по мнению автора, следует исходить при разработке программы школьного курса. А в том, что новая программа необходима, автор убежден.

1. *Базы данных и базы знаний.* Необходимо не акцентировать внимание на принципах их разработки (хотя такие знания, пожалуй, нужны), а уделять больше внимания работе с ними: описанию данных, занесению их в базу, получению справок. Эту работу можно вести на всех уроках. Нет ничего зазорного в том, что на уроке истории ученик во время ответа обратится к компьютеру за забытой датой. Ведь именно так мы и поступаем в жизни — с той лишь разницей, что машина избавит нас от длительного поиска в справочниках. На практических занятиях школьники могли бы создавать (используя готовые системы управления базами данных (СУБД)) различные информационно-справочные системы по истории родного края, по учебному материалу других предметов и т. п. Ведь единицы из сегодняшних школьников будут сами разрабатывать СУБД, а пользоваться ими будут многие, и то, что мы научим детей работать с базой данных, им еще не раз пригодится.

2. *Управление технологическими процессами.* Подавляющее большинство тех школьников, которые станут рабочими, будут иметь дело со станками с программным управлением, гибкими автоматизированными производствами и т. п. Так давайте же учить работать с ними! Давайте через практику на предприятиях или другими средствами учить школьников программировать и обслуживать эти станки и линии. Это трудно и дорого, но цель — подготовка целого поколения к активной жизни — способна оправдать любые затраты.

3. *Средства автоматизации конторского труда.* Это в первую очередь системы типа VISICALC, MULTIPLAN и т. д. Вне всякого сомнения, в ближайшем будущем они будут широко применяться во всех отраслях народного хозяйства.

Уже сегодня язык общения пользователя-непрограммиста с ЭВМ — это язык систем, подобных описанным выше. Времена, когда для работы на ЭВМ пользователь должен был уметь программировать, канули в лету, поэтому поголовное обучение школьников языкам типа Бейсик, Рапира и другим нацелено не в будущее, а в прошлое.

В общем-то автор не настаивает на тех темах, которые были перечислены выше. Единственное положение, которое принципиально важно в любой программе обучения информатике,— она должна быть построена так, чтобы учащиеся поняли: «Компьютер — это нечто, облегчающее жизнь людям!»

Э. ВАРТАПЕТЯН

Не повторять старых ошибок

Это письмо адресовано в первую очередь учителям, педагогам-практикам, на чьи плечи ложится основная тяжесть решения проблем компьютеризации образования. Сейчас, когда период первой растерянности миновал, предмет прижился, не за горами оснащение школ необходимым количеством машин, наступает время собраться с мыслями, подумать, принять более обоснованные решения о дальнейшей судьбе школьного компьютера.

Предположим, что...

Привезли, поставили, работает! Замечательный, надежный компьютер, оснащенный современной операционной системой. Да не один компьютер — локальная сеть! Исправная, поддерживаемая обслуживающей организацией, электронщиками, системщиками, консультантами. Поставили даже гигиенический дисплей — чтобы детишки поменьше портили глаза. Ничто больше не мешает ребенку получить, впитать в компьютерном кабинете все то, что задумано взрослыми для его блага.

А дальше что?

Прежде всего научим ребенка, а главное — учителя и родителей, не бояться новой техники, одолеем синдром компьютерной робости. Освоим клавиатуру. Вот уже школьник всякий раз нажимает именно ту клавишу, какую нужно! И даже может послать товарищу на его рабочее место сообщение по локальной сети. А что там, на следующих этапах компьютерной грамотности? Каков их смысл и содержание?

Какие мы преследуем общественные, педагогические, нравственные цели?

Кажется, ответы даны: они содержатся в учебном пособии, заложены в программном обеспечении. Если что не так, подправим — имеем налаженный механизм распростране-

ния, корректировки информации на местах, вплоть до изъятия из обращения устаревших и негодных программных средств...

И все-таки есть над чем подумать. Не превратить бы дорогостоящую машину в электронное табло для старой игры в «морской бой» или в новую логарифмическую линейку, дающую вместо трех верных знаков восемь. Эксцессы компьютеризации других областей нашего народного хозяйства еще живы в памяти профессионалов. Уже не раз стреляли из пушек по воробьям. Лишь недавно перестали думать, что ЭВМ (пусть даже пятого поколения) в состоянии заменить толкового плановика, опытного бухгалтера, секретаря. Не угрожает больше компьютер сокращением преподавательского состава. Поняли мы, что человеку — «человеково», а компьютеру — «компьютерово». Так не будем в школе допускать старых ошибок! Если мы закрываем глаза на финансовые затраты, то ни в коем случае нельзя забывать об эффективности и качестве. Вот почему следует высказать некоторые соображения.

Кто хозяин в школе?

В союзе учителя с программистом незыблемым должно быть правило: лидер, хозяин, заказчик — это педагог-предметник высокой квалификации. Ни программисты, ни электронщики даже высшей квалификации, заморские или доморощенные, не в состоянии решить за учителя, что ему нужно от их ремесла. Между тем зачастую создание обучающих программ ведется при явном доминировании программиста, не знакомого даже с азами педагогической грамотности, над несомненным носителем педагогической культуры — учителем. Это было неизбежной болезнью роста на первом этапе компьютеризации, когда учитель не был даже приблизительно знаком с возможностями и ограничениями компьютера.

Говорить языком педагога, служить его интересам

Программисту гораздо проще оптимизировать свой программный компонент в своих интересах, нежели облегчить труд ученика и учителя. Этим объясняется столь большая разница в уровнях автоматизации программирования и любой другой области. Программист охотно возьмется за очередную оптимизацию очередной версии компилятора, но подчас пренебрежительно и высокомерно отнесется к тяжелой задаче создания программы под руководством педагога.

Необходимо четко отличать обучающие программы от средств их создания. Не следует увлекаться созданием системных программ (это дело программистов); идти на распространение инструментального средства следует только тогда, когда убедительно продемонстрированы созданные с его помощью педагогически эффективные программные средства, готовые к использованию учителем. Поменьше надо говорить с педагогом о версиях операционных систем, трансляторов, систем управления базами данных и т. д. Оценивать их полезность не дело педагога-предметника; полезность же программного продукта, разработанного для школы, может оценить только он.

Кто автор, откуда заказ?

Любопытно, что обучающие программы по программированию, созданные программистами для самих себя, выглядят довольно жалко с методической точки зрения. Лишь немногие квалифицированные программисты умеют готовить себе достойную смену и, заметим, делают это старым «цеховым» способом: мастер — подмастерье, тяжелый труд, индивидуальный подход, годы старания — и без гарантии результата, а под конец — оценка этическими категориями, а не алгоритмом... Где уж тут без математика создать математическую учебную программу, не говоря уж о литературе или географии! Значит, содержание обучающих программ должно тщательно отбираться и контролироваться педагогической общественностью. Нельзя допускать захламления школы бессодержательными, отупляющими, хотя и внешне эффектными обучающими программами. Здесь нужна полная гласность, с публикацией и обсуждением концепций, сценариев, результатов.

Дать возможность выбора

Всем известно, что настоящий педагог,— как правило, яркая творческая индивидуальность, и подтверждение тому — наши новаторы. Поэтому созданная под его руководством обучающая программа неминуемо несет на себе отпечаток и его личности. Следует разрешить финансируемое создание разными педагогами программ на формально одну и ту же тему — пусть учителя на местах сами выберут наиболее подходящую для них программу, сценарий, методику; статистика их использования даст объективную оценку качества и личного вклада каждого новатора в общегосударственное дело (в отличие от обычных материалов тиражирование обучающих программ можно организовать довольно эффективно).

Аторская ответственность взамен бумаг

Проблема сопровождения программных средств авторской организацией — одна из наиболее сложных в жизненном цикле программного изделия. Сопровождение — это горы разного рода нормативных документов, отчетов, заключений, комментариев и справок. Однако вспомним, что авторский

коллектив, написавший обычный учебник, успешно справлялся с проблемой его поддержки: исправлял ошибки, реагировал на замечания, выполнял требования типографии. Быть может, не всегда следует требовать непременного «отчуждения» продукта труда от авторов ценой горы бумаг — особенно это касается методических решений, которые вообще слабо поддаются формализации и фиксации в документах. Достаточно сделать программы именными, с возможностью конкуренции — со всеми вытекающими отсюда финансовыми и правовыми последствиями. Нередко вмешательство коллектива «сопровождения» приводит к коллективной безответственности. Это не касается, впрочем, системных программ заводского производства — о них мы договорились говорить поменьше.

Компьютерная и общая культура

Когда некоторый предмет переходит из сферы профессиональной деятельности в область массовой грамотности, он неминуемо должен претерпевать серьезные изменения. Заново осмысливаются основные категории науки, расстановка акцентов, система обозначений. Программисты должны осторожно относиться к перенесению своего ремесла в школу. Не стоит удивляться использованию в программировании обозначений на родных языках, а не только на аглийском, не стоит обижаться, что алгоритмические находки в операционной системе будут запрятаны на невидимую школьнику глубину. Не страшно, что из нескольких тысяч терминов выживает лишь сотня — нужно посмотреть в лицо действительности и помочь школе сделать правильный выбор. И если для пересылки сообщения с рабочего места учителя на рабочее место ученика нужно набрать на клавиатуре непроизносимое сочетание английских и русских сокращений, а для того чтобы посмотреть нужный фрагмент текста, затеять целый диалог с устройствами, то следует смело отказаться от плохой операционной системы и потребовать, чтобы простые вещи и делались просто. Здесь профессиональная культура должна неизменно выверяться общечеловеческой, вековой традицией. Мнение программистов о том, что безразлично, каким образом обозначать действия и объекты в компьютере, чревато воспитанием «шаманства», терпеливого и неосмысленного ведения алогичной деятельности. Не все равно, как называть файлы, как адресовать рабочие места, как и из каких слов составлять программу.

В последнее время нередко приходится слышать требования «гуманитаризации» образования, учета исторических и культурных традиций. Это не значит, что часы информатики нужно отдать на другие предметы. Важно другое: так построить курс, находить такие примеры, из которых будет видно нужное для человека применение машины.

Это последнее соображение адресовано больше программистам. Врач, не понимающий по латыни, не может считаться врачом, но правила употребления лекарств, сведения по гигиене и санитарии, как и названия биологических видов, нужно знать по-русски.

А. МИХАЙЛОВ

Диалог Учителя с Программистом, или Что такое информатика

— В школе появился предмет «информатика». Как следует понимать это слово?

— Не знаю. Можно изобрести сколько угодно новых слов, но не нужно думать, что от одного этого в них появляется смысл.

— Есть ли разница между информатикой и программированием?

— На практике — нет. По существу, нынешний учебник информатики представляет собой неудачный курс элементарных основ программирования.

— Чем именно неудачный?

— Недостатки его слишком многочисленны, чтобы имело смысл их перечислять. Укажу один, который можно заметить, прочтя только оглавление. Так называемый алгоритмический язык, неизвестно зачем придуманный, занимает в общей сложности 105 страниц, доморощенная Рапира — 14, а всемирно распространенный Бейсик — 7.

— Но, очевидно, это сделано для повышения доступности материала.

— Никто не требует сделать, например, бином Ньютона доступным дошкольникам — это не тот предмет, который можно изложить на уровне детского лепета. То же и с

программированием. Попытка низвести программирование на вышеуказанный уровень — иначе, боже упаси, наши старшеклассники могут не понять! — дала неизбежный результат: программирование, став детским лепетом, перестало быть программированием, несмотря на именитость авторов учебника. Академические круги весьма удалены от средней школы и не всегда правильно понимают ее проблемы.

— Ну хорошо, а чем плоха «доморощенная», как вы выражаетесь, Рапира? Она — часть «Школьницы», системы, разработанной в Новосибирске специально для учебных целей с программированием на русском языке.

— Простите, вы знакомы с мировым уровнем школьной компьютеризации и с соответствующим программным обеспечением?

— Пока нет.

— К сожалению. Я бы не хотел показаться плохим патриотом, но факт есть факт: наша страна не занимает передовых позиций в школьной компьютеризации. Это само по себе печально, но гораздо печальнее другое. Вместо изучения и использования уже накопленного опыта с усердием, достойным лучшего применения, заново открывают все Америки и самостоятельно изобретают все велосипеды. Львиная доля учителей про компьютеризацию за рубежом ничего не знает. Появилась «Школьница», и на безрыбье она кажется чуть ли не китом. Отсюда же русский язык — ясно, что так «патриотичнее».

— Но, может быть, незнакомая терминология затрудняет изучение предмета?

— Любой школьник знает треугольник «АВС», а не треугольник «АБВ» — в математике пользуются латинскими буквами. Точно так же в программировании используются английские (американские, если угодно) термины. А «затруднение» состоит в том, что ученик должен за два года выучить примерно сотню английских слов. Если для кого-то это — существенное препятствие, то такой случай заслуживает внимания уже не педагогов, а врачей. Для сравнения: дети, обучающиеся музыке, в гораздо более раннем возрасте отлично понимают *итальянские* слова *forte, piano, allegro* и т. п.

— Хорошо, допустим; а как быть учителю?

— Учителя бывают разные. Некоторые, например, больше всего боятся что-либо нарушить: отклониться от учебника, не соблюсти методические указания и т. п. Что при этом усваивают ученики, их не волнует.

— Не стоит излишне заострять внимание на наших недостатках. Будем оптимистами: учитель желает добиться наилучшего конечного результата. Как ему действовать?

— В машинном варианте или в безмашинном?

— Допустим, в безмашинном.

— Тогда не знаю. Нельзя играть в футбол без мяча, нельзя и программировать без машины.

— Ну, а в машинном?

— Поскольку нет разницы между информатикой и элементарным программированием, учителю в первую очередь надлежит стать программистом какой-то минимально необходимой квалификации.

— Как можно это сделать быстро?

— Никак.

— Но позвольте, учитель — это не ученик. Он старше и умнее, у него есть высшее образование и опыт работы, это не может не дать каких-то преимуществ.

— Вспомните слова мудреца: *«В геометрии нет царского пути»*. Чины, звания и заслуги в прошлом не играют *никакой* роли. Можно вспомнить конец 50-х — начало 60-х гг., когда началась «большая» компьютеризация (приличная по тем временам машина занимала чуть ли не этаж). Тогда было не редкость услышать из уст старого, опытного, заслуженного работника какую-нибудь околесицу вроде: «А зачем портят перфокарты, прокалывая в них дырочки?»

— И с тех пор ничего не изменилось?

— Изменилось многое, но только не печальное сочетание невежества и самоуверенности. Вот некоторые случаи из жизни. «Сколько нужно времени, чтобы научиться работать на машине?» (с наивной уверенностью, что этому любой может научиться). «Покажите, пожалуйста, программу» (задавший вопрос полагал, что программа — это материальный предмет). «Почему дискета квадратная?»

— Это говорили дети или взрослые?

— В основном взрослые.

— Тогда понятно: все новое дети усваивают лучше.

— Применительно к информатике ваш тезис станет верным, если поставить его с головы на ноги. Не дети усваивают информатику лучше взрослых, а взрослые *хуже* детей.

— Почему?

— Потому что школьник имеет два крупных преимущества: во-первых, чисто механическую привычку учиться (с возрастом она, как правило, утрачивается), во-вторых, он знает, что изучаемый предмет превосходит (пока) его разумение (а взрослый обычно убежден в обратном). Если школьник чего-то не знает или недопонимает, для него это нормальное явление — с помощью прилежания и усидчивости он может

добиться лучших результатов. Великовозрастные же ученики, порой сами этого не подозревая, твердо верят: «Я старше — значит, умнее. Да и что тут сложного? По телевизору я каждый день футбол смотрю, на машинке тоже печатать умею, значит, как только сяду за пульт, так сразу и начну работать».

— И это им удается?

— Разумеется, нет. И, что самое печальное, получается типичная ситуация «мартышка и очки».

— А бывают ли взрослые ученики без этого комплекса превосходства?

— Крайне редко.

— И как они усваивают материал?

— Вполне нормально. Но, еще раз повторяю, это единичные случаи.

— Как бы то ни было, «идеальный» ученик все же существует. Каков его образ действий?

— Прежде всего он не рыщет по лабиринтам, не занимается стрельбой и бомбометанием, не участвует в автогонках...

— Вы имеете в виду электронные игры?

— Да, именно это нелепое и бессмысленное занятие, чем-то смахивающее на наркоманию,— отсюда и ходячее мнение об их полезности: наркоман, конечно, хвалит наркотик.

— Давайте перейдем к позитивным утверждениям. Что хороший ученик должен делать?

— Он должен учиться читать, писать и считать.

— В самом деле?

— Да, именно так, в полном соответствии с выражением «компьютерная грамотность». Собственно, по этой причине и возникает преимущество детей перед взрослыми — последние считают для себя слишком унизительным заниматься подобной элементарщиной.

— Тогда давайте разберем все по отдельности. «Читать», вероятно, означает «понимать текст на экране дисплея»?

— Читать — значит смотреть на экран и понимать происходящие там процессы. Это не имеет ничего общего с обычным представлением о чтении — слева направо и сверху вниз. Экран читается, как правило, с левого нижнего угла, хотя и не обязательно; глазу порой приходится описывать весьма причудливые траектории. Я уж не говорю о том, что надо знать предмет. Например, один известный мне пользователь при виде сообщения I/O ERROR радостно восклицал: «Вот, единица, деленная на ноль,— ошибка!» Увы, он не знал слов INPUT и OUTPUT, а они должны быть так же понятны, как «мама» и «папа».

— Вероятно, чтобы понимать эти и другие слова, надо внимательно изучить документацию к машине?

— Только перед этим нужно еще прочитать какой-нибудь приличный учебник по программированию — документация рассчитана на компетентного пользователя. В противном случае возникает опасное непонимание высшего порядка. В придачу к тому, что пользователь не понимает инструкций, он еще и не осознает собственного невежества.

— Но, допустим, нашелся трудолюбивый пользователь, который прочитал и запомнил все, что нужно. Может ли он после этого свободно работать на машине?

— Нет и еще раз нет. Никакая инструкция не может дать ответ на вопрос, что происходит в машине в данную секунду. До этого нужно дойти самостоятельно; помнить все наизусть и уметь применять абстрактные знания в конкретных ситуациях.

— Давайте пока кончим с чтением. Остались еще письмо и счет. В чем состоит умение «писать»?

— Разумеется, не в том, чтобы печатать на пишущей машинке. Клавиатура ЭВМ сложнее, чем у машинки (это видно и невооруженным глазом), а экран дисплея отнюдь не лист бумаги. Нажатие клавиши инициирует длиннейшую последовательность событий, в которой иногда присутствует появление символа на экране, а иногда — нет. Есть некоторая аналогия между пользователем ЭВМ и шофером. Последний крутит руль, дергает рычаги и давит на педали; но суть не в том, что он манипулирует этими предметами, а в том, что он делает это *правильно* — и ведет автомобиль. Точно так же пользователь управляет электронной машиной, нажимая те или иные клавиши, и он должен это делать *правильно*.

— А что вы скажете про счет?

— Машинная математика мало похожа на обычную. Надо уметь считать не только в привычной нам десятичной системе счисления, но также в двоичной и шестнадцатеричной. Надо уметь считать биты и байты, строки и символы. Надо знать, что 1 и 1.0 — разные вещи, что $3 \cdot (1/3)$ сплошь и рядом не равно 1, а $(\sqrt{2})^2$ не равно 2... В общем, чтобы завершить затянувшуюся беседу, скажу вам то, что много раз говорил другим: «Мы не в детском саду, я не воспитательница и читать книгу вслух не собираюсь». Такой подход разделял всех моих собеседников на большую и лучшую части. Первые оскорблялись и даже, бывало, жаловались, вторые спрашивали, что нужно прочесть. Куда вы отнесете себя — решайте сами.

Нам пишут

Не откладывая на будущее

В настоящее время (из-за острого дефицита вычислительной техники) в школах страны как базовый реализуется безмашинный вариант изучения курса основ информатики и вычислительной техники. По эффективности этот вариант вряд ли можно даже сравнивать с вариантом, основанным на поддержке теоретического курса практикумом на ЭВМ. Скорейшее и широкое внедрение машинного варианта курса ОИВТ необходимо.

В ближайшие несколько лет дефицит учебной микропроцессорной техники еще будет существовать. Но несколько лет — это целое поколение приходящих на производство и в вузы выпускников школ, СПТУ, техникумов. Можно ли обеспечить более высокий уровень их компьютерной грамотности уже сейчас?

Городское объединение по информатике и вычислительной технике (ГОИВТ), созданное в мае 1985 года и объединяющее представителей вузов, предприятий, народного образования, проанализировав проблемы компьютеризации образования, разработало проект их решения в городе Омске, в основе которого — создание районных компьютерных центров. В учебных целях следует использовать технику, принадлежащую предприятиям города.

Проект поддержали ГК КПСС и горисполком. Соответствующие задания были даны руководителям районов. В ГК КПСС состоялось утверждение районных планов компьютеризации средних учебных заведений и в марте 1986 года было принято совместное постановление ГК КПСС и горисполкома. Согласно этому постановлению, созданы и уже работают центры информатики в четырех районах города. Часть школ других районов прикреплена к ВЦ предприятий-шефов.

В Омске, как, вероятно, и во многих других городах, ряд предприятий, владеющих вычислительной техникой, испытывают острую потребность в площадях. Исполкомы, внимательно изучив ситуацию, нашли возможным заключить с этими предприятиями договоры и сдать им в аренду помещения (бывшие здания школ и учреждений). В свою очередь предприятия оборудовали дисплейные классы, в которых все десятиклассники района проходят практикум по ОИВТ. Как правило, это ЭВМ серии ЕС или СМ-4. Работники предприятий осуществляют обслуживание и ремонт техники, а также системную поддержку. Таким образом, районные компьютерные центры создаются на взаимовыгодных условиях, что позволяет преодолевать ведомственные барьеры. Для координации усилий различных организаций по вводу в эксплуатацию районных центров ВТ создаются штабы под руководством секретаря райкома партии или заместителя председателя райисполкома. К сентябрю 1987 года мы предоставим возможность всем десятиклассникам города, значительной части учащихся СПТУ и техникумов осваивать машинный вариант курса ОИВТ.

Несколько слов об организации изучения курса ОИВТ в омских школах. Девятиклассники, как правило, изучают этот предмет в безмашинном варианте. Для десятиклассников практикум на ЭВМ (4 ч в месяц) организован в районных компьютерных центрах под руководством квалифицированных преподавателей. Расписание занятий координируется районо.

Подготовку преподавательских кадров для школы на краткосрочных курсах при институте усовершенствования учителей нельзя признать эффективной. В Омске школьный учитель в течение всего учебного года посещает компьютерный центр, изучает ЭВМ и методику преподавания, осваивается с обстановкой компьютерного класса. Через год-два он, как только в его школе будет установлена вычислительная техника, сможет самостоятельно преподавать весь курс.

Как известно, эффективным средством активизации познавательной деятельности учащихся являются предметные олимпиады. В Омске разработан комплекс олимпиад по информатике. На командную городскую олимпиаду каждая школа представляет одну команду девяти-, десятиклассников. В заданиях отражены наиболее важные темы курса, поэтому такую олимпиаду можно рассматривать и как итоговую контрольную по ОИВТ. Кроме того, проводится ставшая уже традиционной командная олимпиада по программированию для «профессионалов», т. е. школьников,

овладевающих профессией программиста на УПК и в кружках при вузах. Командная форма олимпиад выбрана не случайно. Такая форма в наибольшей степени стимулирует работу не только учащихся, но и учителей, потому что успех команды — бесспорное достижение учителя, объективное отражение уровня преподавания. В компьютерных центрах, в кружках при вузах учащиеся, овладевающие программированием, осуществляют серьезные самостоятельные разработки.

В сельских школах области также проводится очно-заочная олимпиада по информатике. Победители первого (заочного) тура приглашаются на весенние каникулы в Омск, где на базе компьютерного центра в течение нескольких дней знакомятся с работой на ЭВМ. После этого второй тур проходит уже с применением ЭВМ. В село выезжает компьютерный класс на колесах.

Опыт внедрения машинного варианта курса ОИВТ в городе Омске подсказывает решение этой задачи в сельских школах. Установка одного компьютерного класса в райцентре позволит всем десятиклассникам района изучать курс ОИВТ в машинном варианте. Особых трудностей с доставкой школьников дважды в месяц из села в райцентр не возникает.

В заключение хочется подчеркнуть, что Омск — обычный областной промышленный центр. Наш опыт может быть полезен многим городам страны. Высокий уровень компьютерной грамотности сотен тысяч выпускников средних учебных заведений — цель, ради которой стоит приложить максимум усилий.

Г. ФРИДМАН

Вышел на маршрут

Методический кабинет создан и действует при управлении Дальневосточной железной дороги. Совместно с железнодорожным и педагогическим институтами впервые организовано выполнение лабораторных работ учащимися десятых классов железнодорожных школ Хабаровского и Приморского краев. Для этой цели используется вагон-лаборатория вычислительной техники. По разработанному методическим кабинетом графику после согласования с соответствующими службами управления Дальневосточной железной дороги вагон-лаборатория вышел на маршрут.

Пребывание вагона-лаборатории, оснащенного профориентационно-пропагандистским материалом, на станциях позволило использовать его и для организации экскурсий. Более 1000 учащихся VII—IX классов, 65 педагогов, жители железнодорожных станций познакомились с техническими возможностями дисплейной техники, проверили свои способности в диалоге с машиной, узнали о Хабаровском институте, где готовят специалистов железнодорожных профессий.

Опыт проведения практических занятий по информатике в лаборатории на колесах заинтересовал работников Амурского, Иркутского и Новосибирского облоно, которые с большим интересом изучали организацию занятий, беседовали с учителями и учащимися, наблюдали их работу за дисплеями, знакомились с содержанием выполняемых десятиклассниками лабораторных работ.

Г. СЕЛИВАНЕНКО
г. Хабаровск

Оформление кабинета информатики

Я полагаю, что кабинет информатики и дисплейный — это два отдельных класса, которые дополняют друг друга. Ясно, что кабинет информатики должен выглядеть необычно. У нас вариантов оформления было несколько.

С чего же начинать? С оборудования места преподавателя. Рядом с ним мы расположили графопроектор. Изображение проецируется на экран. Встроили плакатницу с электрическими схемами микрокалькулятора, блок-схемами ЭВМ. Изготовили стенды: «Новое в технике» с вырезками из газет и журналов, «Пятилетний план экономического и социального развития на 1986—1990 годы», «Элементы вычислительной техники». Есть в нашем кабинете движущийся транспарант, где написано решение примера на программируемом микрокалькуляторе. Учащиеся работают на этих микрокалькуляторах, которые включаются с пульта управления преподавателя. На задней стене кабинета стенд — «История развития вычислительной техники». Есть у нас и действующий макет «Перевод из десятичной системы счисления в двоичную». Макет изготовлен на занятиях технического кружка. Сейчас кружковцы разрабатывают действующий макет-тренажер «Пульт управления ЭВМ М-6000». На стене развешаны крупного формата фотографии. Каждый месяц в кабинете ИВТ прибавляется новый стенд или учебное пособие. А это делает оформление кабинета лучше, а занятия, проводимые здесь, интересней.

Г. ИСАЕВ,
преподаватель СПТУ-218, г. Кингисепп

Нужен заведующий кабинетом

Хотим привлечь внимание наших читателей к такому вопросу.

С появлением нового предмета — ОИ и ВТ — в областных институтах усовершенствования учителей введена ставка методиста, который, по существу, является и заведующим кабинетом информатики.

Ему приходится отвечать за сохранность техники, за поддержание машин в рабочем состоянии.

Уезжая в область проверять школы, методист закрывает кабинет на ключ, а ключ, понятно, кладет себе в карман. И он прав — отвечать за компьютеры ему! Вернувшись, он хочет найти свой КУВТ в целости. А вот по-хозяйски ли это, когда простаивает дорогая техника? За оформление кабинета тоже отвечает методист. Сам все достает, придумывает всевозможные приспособления, проявляет чудеса изобретательности. Это помимо основных обязанностей.

А основное в его работе — обеспечить повышение квалификации учителей. В Саратовской области, например, 675 школ с преподаванием информатики. Кроме того, 3 тыс. учителей других предметов ежегодно проводят переподготовку в институте. Их тоже необходимо познакомить с компьютерами, а это можно сделать только в кабинете информатики. Преподавателям информатики техникумов и СПТУ, желающим поработать с ЭВМ, тоже не откажешь. (В Саратовской области их 130.)

И вот сегодня всем этим занимается один человек: помогает учителю практически осваивать вычислительную технику; обеспечивает его программами, которые сам собирает, хранит, тиражирует, распространяет передовой педагогический опыт. И еще много ежедневных текущих дел...

Пора помимо должности методиста ввести дополнительно хотя бы ставку заведующего кабинетом информатики!

Компьютерная инквизиция

Церковь недаром считается организацией консервативной, но и Ватикан использует в своей деятельности ЭВМ. К декабрю 1987 г. получит свою локальную сеть Конгрегация по вопросам вероучения — организация, призванная выявлять богословские ошибки, иначе говоря, ересь. Особая забота при этом уделяется мерам по обеспечению секретности программ и данных — каждый из устанавливаемых ПК сможет работать автономно; сеть не будет иметь связей с внешним миром.

Компьютеры используются и в финансовой деятельности католической церкви, и в ее Центральном статистическом управлении. Планируется их установка в редакции газеты «Оссерваторе романо» и в многоязычной ватиканской типографии.

ЧТО МОЖЕТ ЭВМ?

Если не сбросит лошадь

Тут уж просто непонятно, как и реагировать. Оказывается, ЭВМ может... скакать! Как заправский жокей! Впрочем, даже не «как»: журнал «Штерн» (ФРГ) сообщил, что в США запатентован электронный жокей. Он управляет лошадью, подает ей команды голосом и сам воспринимает команды хозяина посредством дистанционного управления.

Предназначен этот «жокей» для скачек, т. е. должен ехать не в коляске, а верхом, так что его пристегивают к лошади ремнями — примерно как седло. Трудно сказать, допустят ли электронного наездника до участия в соревнованиях, а если допустят... Хорошая скаковая лошадь — существо нервное, не оценит новинку — ремни могут и не выдержать...

Совещание в Коломенском пединституте

4—5 июня 1987 г. на базе Коломенского педагогического института проходило рабочее совещание разработчиков инструментальных программных средств для КУВТ-86; рассматривались состояние и перспективы развития инструментального программного обеспечения, демонстрировались различные программные средства для КУВТ-86.

Все участники отметили недостаточность технических возможностей КУВТ-86 для решения инструментальных задач, например для разработки баз данных. В то же время наличие большого числа этих комплектов в учебных заведениях страны делает необходимой интенсивную разработку педагогических и инструментальных программных средств.

В настоящее время пользователи имеют три не полностью программно совместимые модификации КУВТ-86:

с пленочной клавиатурой и языком Фокал;

с полноходовой (кнопочной) клавиатурой и языком Фокал;

с полноходовой клавиатурой и языком Бейсик (поставки начались с 1.04.87 г.).

Представители организаций, получивших комплект с Бейсиком, рассказали об ошибках в интерпретаторе Бейсика и о плохой реализации локальной сети.

Совещание выработало рекомендации по разработке программного обеспечения:

к первоочередным задачам разработки инструментальных программных средств отнести текстовый, графический и музыкальный редакторы, синтезатор звука, электронные таблицы, учебные базы данных, генератор задач, конструктор урока;

разработку инструментальных программных средств для КУВТ-86 по возможности производить на инструментальных вычислительных машинах (СМ-4, СМ1420 и др.);

поставки комплектов КУВТ-86 с Бейсиком в ПЗУ необходимо приостановить для устранения замеченных ошибок и разработки программного обеспечения;

разработать с ПО «Экситон» план мероприятий по доработке интерпретатора Бейсика и локальной сети;

сформировать группу на базе ЦКО ИАЭ им. И. В. Курчатова по выработке соглашений для обеспечения унификации инструментальных программных средств по использованию клавиатуры, распайке печатающих устройств, программированию драйверов, программной поддержке локальных сетей, оформлению программных средств.

Некоторые соглашения удалось сформулировать в процессе совещания.

1. Программа должна включать в себя краткую инструкцию по ее использованию. В случае необходимости более подробная инструкция по работе с программой может быть вынесена в отдельный текстовой файл.

2. Заголовок программы должен содержать информацию об организации или авторе разработки с указанием адреса или телефона.

3. Предлагается осуществлять подключение печатающего устройства и разработку соответствующей программной поддержки с учетом схемы в приложении 1.

4. Предлагается набор команд управления экранным редактированием текстов (выбор сделан, исходя из набора клавиш БК-0010 с полноходовой клавиатурой, который является минимальным).

↑↓→←	клавиши управления перемещением курсора
|→	раздвинуть строку
|←	сдвинуть строку
→|	сброс строки справа от курсора
←|	удаление последнего введенного символа
ВВОД	переход в начало следующей строки
ВС	переход в начало текущей страницы
СБР	сброс страницы (требует подтверждения)
КТ,↓	следующая страница*
КТ,↑	предыдущая страница
КТ, ?	справочная информация
КТ, СБР	сброс всего текста (требует подтверждения)
КТ, N(H)	в начало текста
КТ, K(K)	в конец текста
КТ, Ч(→)	чтение файла
КТ, З(Z)	запись файла
КТ, П(P)	печать файла
КТ, И(I)	искать (например, поиск метки, контекста и т. п.)
КТ, P(R)	размножить (например, строку)

*Клавише КТ полноходовой клавиатуры соответствует пустая голубая кнопка в верхнем ряду на пленочной клавиатуре.

5. Выбор командных клавиш для ПС других типов предлагается осуществлять в рамках предложенной для экранного редактора идеологии.

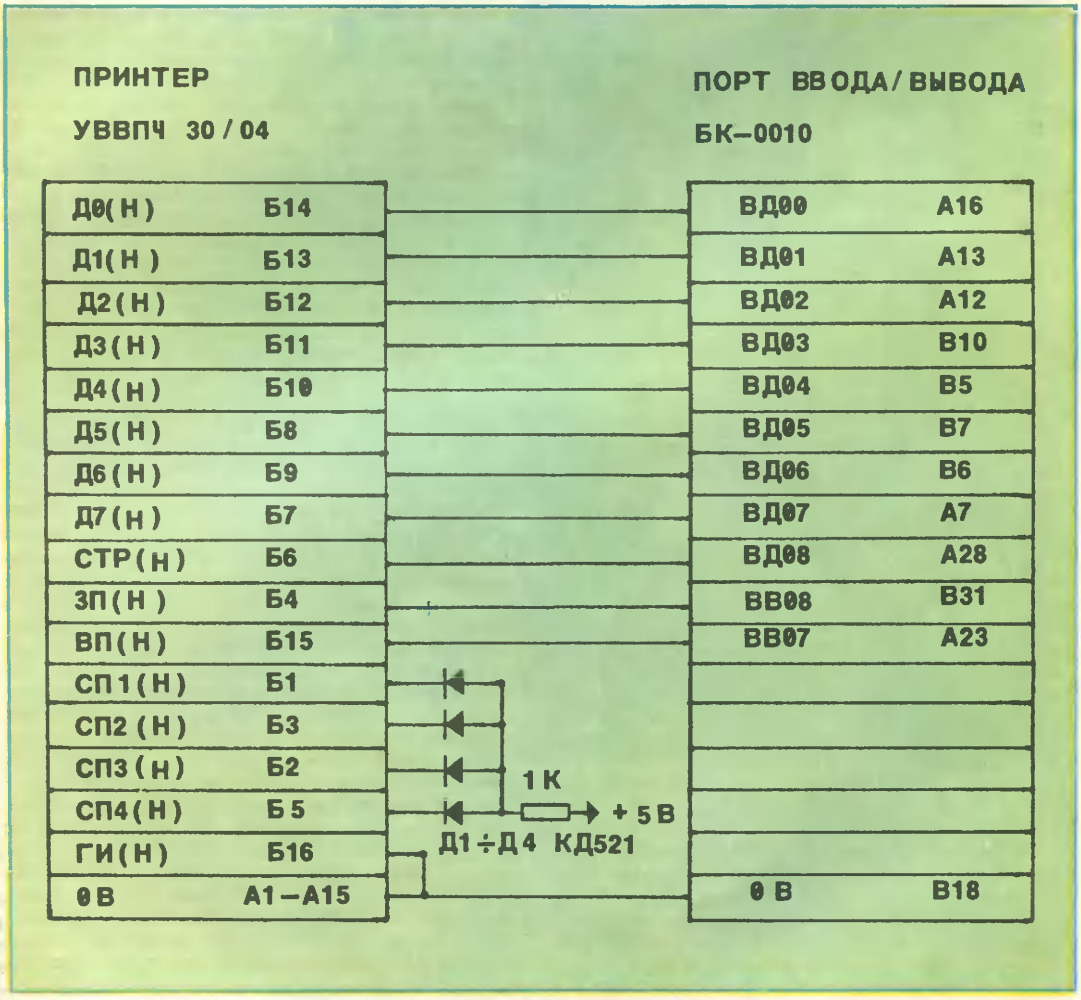
Выполнение программистом рекомендаций и соглашений будет способствовать широкому распространению его программы, обеспечит ее работоспособность на всех модификациях БК-0010 и для пользователя любого уровня подготовки сократит время обучения, необходимое для работы с ней.

Участники совещания выработали план разработки некоторых инструментальных программ, констатировали необходимость проведения периодических рабочих совещаний-семинаров по данной тематике.

План разработки инструментальных программных средств для КУВТ-86

Организация	Наименование программного средства	Срок выполнения
ЦКО ИАЭ им. И. В. Курчатова	Интерактивный графический редактор	По выработке соглашений
Студенческий научно-производственный отряд МФТИ	Кольцевая локальная сеть. Текстовый и графический редакторы для БК—0010	Сентябрь 1987 г.
ЦНИИ «Электроника»	СУБД РДБ-микро	1988 г.
Орехово-Зуевский пединститут	Генератор задач. Авторский язык для написания структурированных ППС	1988—1989 гг.
Коломенский пединститут	Конструктор урока	1988 г.

Схема кабеля печати БК-0010Ш с Бейсиком



Необходимость соглашений между разработчиками программных средств давно назрела. В ближайшее время необходимо выработать соглашения по использованию контактов порта ввода/вывода для программирования внешних устройств («мышь», выносной пульт и др.), да и рекомендации совещания могут вызвать споры. Если у Вас есть конкретные предложения по этой теме, присылайте их в редакцию журнала. Не забудьте указать свой адрес и телефон!

Всесоюзный семинар «Использование КУВТ-86 в учебном процессе»

Школа ждет вычислительную технику. И учителя и ученики убеждены, что появление компьютеров поможет лучше усвоить учебный материал. И их надежды начинают сбываться: более тысячи первых отечественных комплексов учебной вычислительной техники КУВТ-86 получили школы страны. Но вместе с техникой возникают и проблемы. Одна из них — подготовка учителей к работе в новых условиях.

Решение этой задачи невозможно без систематического обмена опытом. Министерством просвещения СССР запланировано проведение серии всесоюзных семинаров по обмену опытом преподавания информатики с использованием педагогических программных средств, на которых педагоги, специалисты смогут не только познакомиться с опытом коллег, записать программное обеспечение, но и получить необходимую консультацию по его использованию, чтобы провести аналогичный семинар у себя в республике.

Всесоюзный семинар «Использование КУВТ-86 в учебном процессе» проходил 9—13 июня 1987 г. в Центре компьютерного обучения (ЦКО) Ворошиловского района Москвы. Во время его работы слушатели подробно ознакомились с техническими возможностями отечественной вычислительной техники, поступающей в школы, получили подробную консультацию старшего инженера НИИ ШОТСО АПН СССР А. Г. Архангельского по вопросам работы с операционной и файловой системами, средствами создания текстов на ЭВМ ДВК-2М. Участники семинара встретились с разработчиками программного обеспечения (заведующим лабораторией НИИ физики твердого тела Латвийского университета Ю. Я. Кузьминым, доцентом механико-математического факультета МГУ А. Г. Кушниренко). Оживленно прошла беседа с сотрудниками завода-изготовителя КУВТ-86. Директор центра компьютерного обучения М. В. Галицкий поделился опытом организации учебного процесса, методики преподавания курса информатики, эксплуатации и обслуживания вычислительной техники. Большой интерес вызвали многие технические решения, используемые в ЦКО: организация сети БК-БК с магнитофоном в качестве внешней памяти; подключение печатающего устройства к ЭВМ БК-0010; пульт, с помощью которого с экрана учительской машины можно проследить за работой учеников, не отвлекая их внимания; использование БК-0010 с демонстрационным телевизором.

Составляя программу занятий, организаторы стремились придать семинару деловой, практический характер. Во второй половине каждого рабочего дня проводились практические занятия в кабинетах вычислительной техники (КУВТ-86, ДВК-2М, БК-0010), где каждый участник имел возможность под руководством специалистов закрепить знания, полученные на лекциях, ознакомиться с различными вариантами организации локальной сети, получить консультацию разработчиков программного обеспечения по использованию в учебном процессе различных программных средств. Вели практические занятия Ю. Я. Кузьмин, С. Л. Яновский (ЦКО), В. В. Касаткина (НИИ ИВТ АПН СССР), Л. А. Андерсоне (Рижский институт усовершенствования учителей Латвийской ССР), А. Г. Архангельский.

Более ста программ различного назначения получили слушатели семинара. В состав программного обеспечения вошли: системные файлы, текстовые редакторы, подготовленные А. Г. Архангельским и инженером ЦКО Е. А. Житницким (диск № 1), пособие по работе с операционной системой RT-11, написанное в доступной для начинающего пользователя форме А. Г. Архангельским (диски № 8, 9), авторская система «Рига», о которой уже сообщалось в нашем журнале (1987, № 2). Были представлены различные программы, разработанные в этой системе под руководством Ю. Я. Кузьмина, в том числе электронный блокнот, исполнитель «Робот», серия программ, обучающих языку программирования Бейсик, и другие (диски № 4, 5, 6). Обучающие программы по информатике и другим предметам, принятые комиссией республиканского фонда алгоритмов и программ Латвийской ССР, разработаны в лаборатории проблем школьной информатики вычислительного центра Латвийского университета (диски № 2, 3). Сотрудниками НИИ ИВТ АПН СССР подготовлен диск № 7, в составе которого — учебная база данных, учебные электронные таблицы, графический редактор, примеры обучающих программ и тренажеров по предметам учебного цикла. Полученное программное обеспечение поможет учителям на уроках полнее использовать возможности вычислительной техники.

В ходе работы семинара поднималось немало проблем, связанных с эксплуатацией КУВТ-86.

Участники семинара отмечали, что ЭВМ ДВК-2М, входящая в состав КУВТ-86 и превышающая по цене половину стоимости всего комплекта, используется учителями только как дисковод из-за ее низкой надежности и сложности в эксплуатации. С комплектом трудно работать еще и потому, что на рабочих местах учителя и ученика установлены несовместимые по программному обеспечению и формам представления информации на экране. В классе БК-БК этой сложности нет, но возникают другие — использование магнитофона требует времени для загрузки программ, качество магнитной ленты не обеспечивает требуемой надежности. Быстро выходит из строя пленочная клавиатура (стирается пленка), вопрос этот можно решить, если организовать дополнительную поставку пленок. Из-за отсутствия термобумаги простаивают термопечатающие устройства. Если обеспечить ею школы так сложно, то, может быть, легче заменить печатающие устройства? Чрезвычайно низкое качество отечественных дисков делает их использование невозможным. Зачем же тогда поставлять их с КУВТ-86? Почему вместо 20 положенных дисков с комплектом приходит зачастую лишь половина, а то и меньше? Эти и другие вопросы задавали разработчикам участники семинара. Пока в школы идут КУВТ-86,

оснащенные языком программирования Фокал, на котором уже разработано программное обеспечение, в том числе и часть полученного слушателями семинара. Но представители завода-изготовителя информировали участников семинара о том, что скоро начнется допоставка ПЗУ с языком Бейсик. Замена ПЗУ означает разработку программного обеспечения практически заново, и хозяевам КУВТ-86 опять придется ждать программы. Лучше было бы, если бы пользователь мог выбирать режим работы (Фокал или Бейсик), переключая ПЗУ.

Минвуз СССР: новый этап в работе

В начале июня 1987 г. в Днепропетровске на базе индустриального техникума было проведено заседание методической комиссии по информатике и электронно-вычислительной технике.

Здесь собрались активные поборники компьютеризации, люди, которым небезразлична судьба этого важнейшего дела.

В выступлениях участников заседания были затронуты крупные вопросы: организация разработки и изготовления ЭВМ в учебно-производственных мастерских техникумов (Г. Маколий, Львовский техникум радиоэлектроники; О. Доминский, Винницкий техникум электронных приборов); методическая работа преподавателя (Е. Сергеева, Минский политехникум и др.); об изучении предмета «Основы информатики и ВТ» в техникумах и училищах в группах на базе средней общеобразовательной школы (А. Коптев, УМУ Минвуза СССР).

Хозяева подготовили исключительно содержательную программу экскурсий по техникумам Днепропетровска; была также показана серия диафильмов, выпускаемых в экспериментальной лаборатории технических средств обучения. Каждому участнику заседания был вручен комплект диафильмов по курсу основ информатики. (О качестве их читатель может судить по прилагаемым иллюстрациям.)

В один из дней была проведена дискуссия о непрерывном компьютерном образовании в средних специальных учебных заведениях.

Присутствующие ознакомились с выставкой методического объединения ВТ Минчермета УССР.

Особый интерес собравшихся специалистов вызвал «круглый стол» журнала «Информатика и образование», посвященный становлению этого издания, столь необходимого преподавателям техникумов.

В течение нескольких часов шел горячий заинтересованный разговор. Открыл встречу А. Фурсенко, член редколлегии журнала. Он представил сотрудников журнала, которые приехали в Днепропетровск, членов авторского актива — директоров и преподавателей ведущих техникумов страны.

В обстоятельном выступлении И. Орешков, директор Рязанского техникума электронных приборов, выразил главную идею — журнал находится на подъеме, он нашел свою читательскую аудиторию, от номера к номеру становится

содержательнее, серьезнее, интереснее. Ю. Молотков (Новосибирск) выразил пожелание, чтобы доля материалов о средних специальных учебных заведениях возросла.

Во многих выступлениях звучала мысль о том, что необходимо давать больше методических материалов, статей о зарубежном опыте, качественных иллюстраций.

Положительная оценка, данная читателями, только аванс, и редакция будет неустанно работать над повышением уровня своих публикаций с учетом всех пожеланий, высказанных на заседании «круглого стола».

АСТРА/МИКРО в Ленинграде

С 17 по 19 июня 1987 г. в Ленинграде на базе Всесоюзного НИИ профтехобразования состоялся научно-практический семинар «Разработка и применение системы АСТРА/МИКРО в учебном процессе» с участием ее разработчиков из Московского института электронной техники.

В работе Всесоюзного семинара приняло участие свыше 100 инженеров, инженерно-педагогических работников, преподавателей, профессорско-преподавательский состав вузов.

На семинаре были продемонстрированы возможности новой версии системы АСТРА/МИКРО, в том числе с использованием машинной графики на базе ДВК-3, локальная сеть на базе микро-ЭВМ ДВК-2 и ДВК-1, музыкальная приставка.

Большой интерес вызвала демонстрация учебных курсов с использованием системы АСТРА/МИКРО.

Из записной книжки программиста

Прежде чем написать программу — убедись, что существует ЭВМ, способная ее переварить.

Чтобы овладеть искусством программирования, нужно в первую очередь научиться владеть собой.

Научись писать коротко сложные программы, и тогда ты сможешь изредка позволить себе писать длинно простые.

Если ты не видишь очевидных ошибок в своей программе, не отчаивайся,— скорее всего, ты близок к тому, чтобы стать настоящим программистом.

Помни, что несколько лишних операторов в программе не принесут такого вреда, как отсутствие двух-трех необходимых.

Если, по-твоему, программа составлена правильно, это еще не значит, что с этим согласится машина.

Помни, что употребление в программе бранных слов и выражений не всегда является действенным способом заставить ее работать.

Если программа заработала, не расстраивайся — долго это продолжаться не будет.

Занимаясь улучшением работающей программы, помни, что это верный способ сделать ее хуже.

Если машина вышла из строя еще до того, как программа стала выдавать результаты,— не унывай,— скорее всего, они были бы ошибочными.

Нет ничего более интересного и загадочного, чем с ошибками составленная, но безошибочно работающая программа.

Вкладывая в программу всю свою душу, не забывай, что ее еще будут эксплуатировать.

Если планируешь написать большую, сверхсложную и, самое главное, безошибочно работающую программу — подумай сначала о своих близких.

Программа — высокая поэзия, результаты ее работы — грубая проза.

Основной вопрос программирования: что сложнее — программа или написавший ее программист?

Каждая программа — отражение достоинств и недостатков ее составителя.

Не разговаривай с друзьями и близкими на алгоритмическом языке — тебя могут неправильно понять.

Утреннее машинное время возьми себе, дневное раздели с другом, а вечернее отдай врагу.

Надпись на дверях машинного зала: «Оставь надежду всяк сюда входящий».

Г. Юдин

Программирование очертя голову

Абсолютно серьезный доклад на совершенно серьезную тему

Уважаемые товарищи, с сожалением приходится констатировать, что широкое распространение вычислительной техники и обучение информатике приводит к учащению заболеваний ПОГ. Но, к счастью, тяжелая форма болезни — редкость. У большинства начинающих программистов ПОГ протекает почти незаметно, не причиняя никаких страданий ни больному, ни окружающим.

Интересно, что у женщин обычно наблюдается легкая форма заболевания, причем женщины старше 40 лет — даже иммунны (по крайней мере у них еще не отмечено ни одного случая заболевания). Опасности заболеть ПОГ в тяжелой форме особенно подвержены юноши переходного возраста, а наиболее серьезные последствия зафиксированы у пожилых мужчин (как и в случае других детских болезней).

Течение болезни ПОГ описывается и измеряется не количеством дней, а числом составленных больным программ. *Инкубационный период* — 3—4 программы. Если после создания 10—20 программ симптомы болезни затухают, можно констатировать легкую форму и, что особенно существенно, приобретение устойчивого иммунитета. Если же внешние признаки ПОГ проявляются и после 20 программ, случай — тяжелый. Остановимся подробнее именно на таком случае.

Симптомы. Первый видимый (или вернее слышимый) признак заболевания проявляется в том, что больной (или ОГ-программист), находясь в состоянии либо эйфории, либо депрессии, непрерывно говорит о своей последней программе. Депрессивное состояние наблюдается, когда программа в очередной раз оказывается неверной, хотя больной по меньшей мере дважды в день объявляет, что неисправна машина. В состоянии эйфории, когда, по мнению ОГ-программиста, текст программы исправлен, больной пытается всем объяснить, в чем же состояла последняя ошибка. Несчастный! Он искренне уверен, что ошибок больше нет! Он еще не понимает, что обречен на бесконечный поиск.

При сложившейся картине болезни в процессе сверхоперативной борьбы с ошибками (известным методом проб и ошибок) программа становится непонятной даже для автора, а сам ОГ-программист теряет в тяжелых случаях способность логического, а в особо тяжелых — любого мышления.

К симптомам ПОГ относится и отсутствие интереса (или потеря его) к литературе по программированию, презрение к системным, структурным, продуманным и дисциплинированным методам программирования. Нередки случаи, когда больной «изобретает колесо».

Стремление при любой возможности демонстрировать уверенную и умелую работу с машиной, головокружительные трюки (от которых нет никакой практической пользы), речь, пестрящая жаргонными словами, такими, как «принтить», «листить» и т. п.,— тоже признаки тяжелой болезни.

У ОГ-программистов наблюдается покраснение глаз, бледность лица, дрожь в руках

---

* **Программирование очертя голову (ПОГ).** *Мед.* — «детская» болезнь, возникающая в начальный период обучения программированию, после того как человек овладевает основами какого-либо языка программирования и составляет первую в своей жизни программу. ПОГ наблюдается у людей различного возраста.

и голосе, особенно когда речь идет об ЭВМ и программировании. В очень тяжелых случаях — разрушение зубов и выпадение волос.

Лечение. Наука еще не знает вполне эффективных средств борьбы с ПОГ. Можно посоветовать хорошо продуманный начальный курс программирования, а также... длительное пребывание на свежем воздухе. Полезно время от времени насильно удалять ОГ-программиста от машины. Например, в подходящий момент вывернуть электрические пробки. Однако необходимо следить, чтобы при этом не пропало слишком много трудов ОГ-программиста, так как его реакции совершенно непредсказуемы. Способность мыслить изолированного от машины больного восстанавливается довольно быстро. Если перед ним каким-то чудом окажется текст программы, в которой он при помощи машины двое суток отчаянно пытался найти ошибку, то вполне вероятно, что теперь он обнаружит ее в течение 10—15 минут.

Благотворно действует на больного знакомство с парой новых языков программирования. Иногда полезно доверить ОГ-программисту большую и сложную задачу. Решая ее, больной скоро поймет, что его знания недостаточны. Это будет способствовать выздоровлению или... обострению болезни.

Люди, страдающие болезнью ПОГ, для общества неопасны. После выздоровления они опять становятся его полноценными членами, а из некоторых выходят даже неплохие программисты.

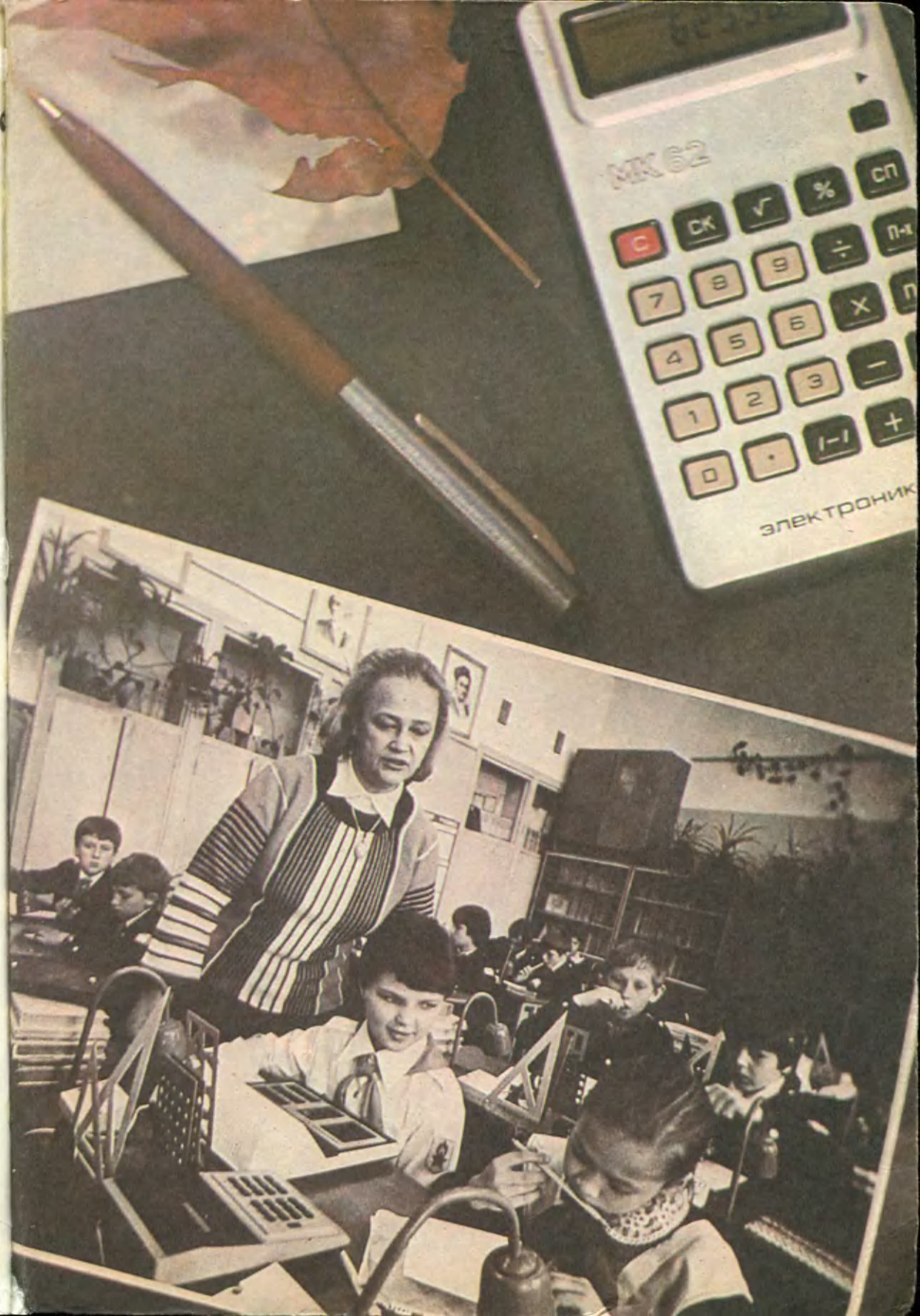
Профилактика. Предупреждение ПОГ крайне затруднительно, особенно у учителей информатики. Более того, ОГ-программист, занимающий эту должность, заражает обучаемых. Учитель, страдающий ПОГ даже в легкой форме, испытывает непреодолимое отвращение к методике обучения программированию и общепризнанным методам обучения. Обычно он учит отдельным командам и трюкачеству с их помощью, а также знакомит со своим богатым опытом на поприще бессистемного поиска ошибок. Он уверен, что программирование — учение о командах языка программирования (наглядно: лесоводство — учение о большом количестве отдельных деревьев).

Распространению ПОГ способствует нехватка учебников и справочников по программированию. До сих пор издательства еще не поняли в полной мере своей ответственности в этом вопросе. В прессе, на радио и по телевидению необходимо ограничить количество выступлений, подстрекающих на ПОГ, при этом увеличив число компетентных выступлений квалифицированных специалистов.

Большие надежды в профилактике болезни возлагаются на профсоюзные, комсомольские и спортивные организации. Несомненно, что при решительной помощи общественности тяжелая форма болезни ПОГ будет окончательно предотвращена.

Ю.-А. КИММ

Докт. медико-программистских наук


МК 62
С
СК
√
%
СП
7
8
9
÷
4
5
6
×
1
2
3
0
+/-
+
ЭЛЕКТРОНИК

Цена 60 коп.

70423

ИНФО

5'87

ИНФОРМАТИКА И ОБРАЗОВАНИЕ