ISSN 0234—0453

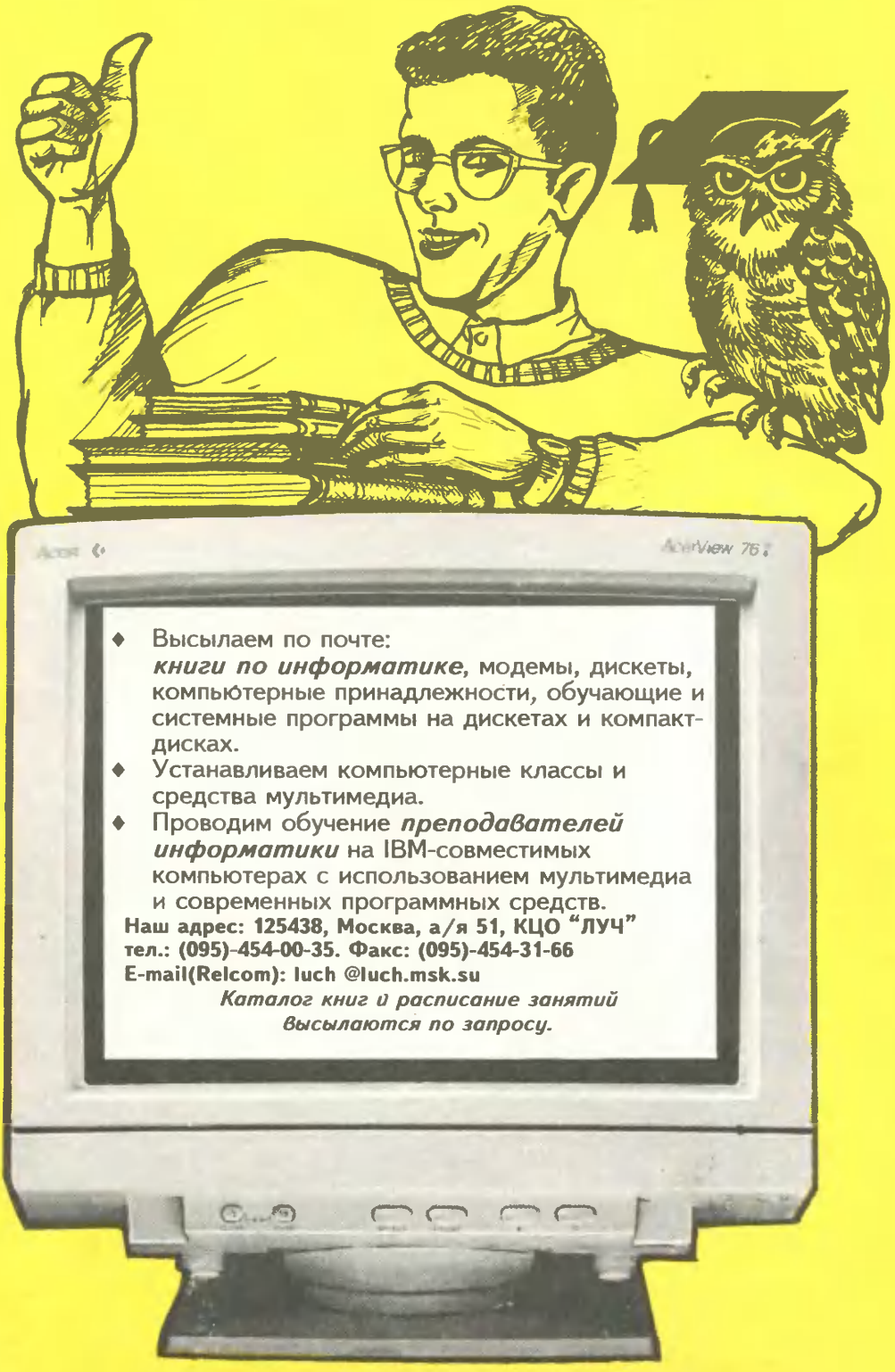
ИНФО

ИНФОРМАТИКА И ОБРАЗОВАНИЕ

1 1995

Компьютерный центр обучения “Луч”

10 лет в авангарде информатизации



- Высылаем по почте:
  книги по информатике, модемы, дискеты, компьютерные принадлежности, обучающие и системные программы на дискетах и компакт-дисках.
- Устанавливаем компьютерные классы и средства мультимедиа.
- Проводим обучение ***преподавателей информатики*** на IBM-совместимых компьютерах с использованием мультимедиа и современных программных средств.

Наш адрес: 125438, Москва, а/я 51, КЦО “ЛУЧ”
тел.: (095)-454-00-35. Факс: (095)-454-31-66
E-mail(Relcom): luch @luch.msk.su

Каталог книг и расписание занятий высылаются по запросу.

ИНФО 1'95

ИНФОРМАТИКА И ОБРАЗОВАНИЕ

Научно-методический журнал
Учрежден Министерством образования РФ и коллективом редакции

Издается с августа 1986 г.
Выходит шесть раз в год

СОДЕРЖАНИЕ

10 ЛЕТ ШКОЛЬНОЙ ИНФОРМАТИКЕ

Ершов А. П., Звенигородский Г. А., Первин Ю. А. Школьная информатика (концепции, состояния, перспективы) 3

РЕГИОНАЛЬНАЯ ПРОГРАММА

Коган Е. Я., Первин Ю. А. Курс «Информационная культура» — региональный компонент школьного образования 21

ОБЩИЕ ВОПРОСЫ

Белкин А. Р., Шумов С. И. Анализ и оценка традиционных и нетрадиционных механизмов получения и обобщения новых знаний *(окончание)* 29

МЕТОДИКА

Залогова Л. А., Семакин И. Г. Базовый курс ОИВТ: «Пермская версия» (часть 4) 41

Шень А. Х. Программирование: теоремы и задачи *(окончание)* 50

Юдина А. Г. Информатика на ЛОГО для старшеклассников *(продолжение)* 57

Маслов А. Н. Для современной школы есть выбор 67

ЗАДАЧИ

Нечаев В. М. Одна задача — три языка 72

Варакин Ю. С. Проекты-задания на языке ЛОГО для учащихся старших классов 86

НОВЫЕ ПРОЕКТЫ

Утлинский Е. В., Аввакумов Э. Г. Телекоммуникационная сеть TIT-BIT *(продолжение)* 93

Богданова Д. А., Федосеев А. А. Дистантное обучение: модератор телеконференций 104

Пронин В. Н. MultiVision Pro — год 1995 109

ПРОФЕССИОНАЛЬНЫЙ КОМПЬЮТЕР

Краснова О. А., Левченко И. В. Излучение компьютера и здоровье детей 113

КЛУБ УКНЦ

Файзуллин В. А. Реализация алгоритма формирования массива с различными случайными элементами 117

КЛУБ «КОРВЕТ»

Яромко Л. И. Партитура в алгоритмических рамках 119

На первой странице обложки — генеральный директор НЦПСО А. А. Зубченко у стенда фирмы CAIRC-AIST group на выставке COMDEX-94 в Лас-Вегасе.

Почту направлять по адресу: 103051, Москва, ул. Садовая-Сухаревская, д. 16, к. 9, журнал «Информатика и образование».

Телефон: (095) 208-30-78
Факс: (095) 208-67-37
E-Mail: info@tit-bit.msk.su

Подписано в печать с оригинал-макета 13.01.95. Формат 70×100 1/16. Бумага офсетная. Печать офсетная. Усл. печ. л. 10,40. Усл. кр.-отт. 11,70. Уч.-изд. л. 13,5. Тираж 12 150 экз. Заказ 3865
Цена по подписке:
для индивидуальных подписчиков 6 000 руб. (индекс 70423);
для предприятий и организаций 15 000 руб. (индекс 73176).
В розницу цена договорная.

Ордена Трудового Красного Знамени Чеховский полиграфический комбинат Комитета по печати и информации Российской Федерации. 142300, Чехов Московской обл.

10 ЛЕТ ШКОЛЬНОЙ ИНФОРМАТИКЕ

А. П. Ершов, Г. А. Звенигородский, Ю. А. Первин

ШКОЛЬНАЯ ИНФОРМАТИКА (концепции, состояния, перспективы)*

Хотя эта статья размещается в номере, посвященном десятилетию отечественной школьной информатики, она старше юбилейной даты. Материал, обсуждавшийся в кабинете Андрея Петровича Ершова в Вычислительном центре Сибирского отделения АН СССР и в киевской гостинице, где мы останавливались с Геннадием Анатольевичем Звенигородским, приобретал вид законченной публикации в последний предновогодний вечер семидесятых. Даже дата на препринте ВЦ СО АН СССР была поставлена «1979». И известный лозунг А. П. Ершова «Программирование — вторая грамотность», и его слова о звездном часе, обращенные к соавторам, которых он напутствовал, командируя в Москву для формирования предложений Госкомитета по науке и технике к намечаемой программе компьютеризации советской школы, были произнесены значительно позднее.

Трудно говорить о чувствах, возникающих сегодня, когда приходится перечитывать страницы препринта: с одной стороны, это неисчерпаемое уважение к коллегам, которые увидели (и помогли увидеть) столь задолго такие удивительные для того времени перспективы, а с другой — скорбная горечь: ни старшему, ни младшему не довелось увидеть наяву то, во что в 1979 г. верили лишь немногие.

В эпоху, когда идеи нового информационного общества несли в школу в основном энтузиасты-программисты, впервые была четко высказана мысль о том, что школьная информатика должна ставить целью вовсе не обучение ремеслу программирования, но формировать стиль мышления у целого поколения людей, вступающих в жизнь. Школьная информатика определялась как ответ школы на социальный заказ современного общества. Несмотря на последующие уточнения и развитие как средств, так и целей информатизации образования, слова, написанные в далеком 1979-м, оказались удивительно актуальными и для своего времени, и для нынешних дней.

Ю. А. Первин

Предмет школьной информатики

В наше время повсеместного распространения электронных вычислительных машин (ЭВМ) человеческие знания о природе информации приобретают общекультурную ценность. Этим объясняется интерес исследователей и практиков всего мира к относительно молодой научной дисциплине — **информатике**, которую в рамках настоящего контекста достаточно определить просто как науку о структуре информации и методах ее обработки на ЭВМ.

При столь широком определении ин-

* Текст печатается с сокращениями по изданию: *Ершов А. П., Звенигородский Г. А., Первин Ю. А.* Школьная информатика (концепции, состояние, перспективы). Препринт. Вып. 152 / Вычислительный центр Сибирского отд-ния Академии наук СССР. Новосибирск. 1979.

форматика содержит в себе **программирование** — дисциплину, исследующую методы формулирования и решения задач с помощью ЭВМ — в качестве составной (хотя несомненно важнейшей, определяющей!) компоненты. Современная информатика постепенно складывается как самостоятельная научная дисциплина со своей теорией. Вместе с тем в каждой из многочисленных и разнообразных прикладных областей, где находят свое применение методы информатики, специфика приложений определяет новые прикладные направления. Одной из таких важнейших областей человеческой деятельности, актуально требующих привлечения новых идей и методов, является система народного образования.

Средства информатики, определяющие ее приложения к проблемам средней школы, формируют прикладную дисциплину — школьную информатику. **Школьная информатика** определяется как ветвь информатики, занимающаяся исследованием и разработкой программного, технического, учебно-методического и организационного обеспечения применения ЭВМ в школьном учебном процессе.

В области технического обеспечения школьная информатика имеет своей целью экономически обосновать выбор технических средств автоматизации учебного процесса; определить параметры оборудования типовых школьных вычислительных центров; найти оптимальное соотношение использования серийных средств и оригинальных разработок, ориентированных на среднюю школу.

Наиболее сложные и трудоемкие проблемы школьной информатики сосредоточены в ее математическом обеспечении, которое представляет собой информационную, управляющую и обучающую систему средней школы, программистский инструментарий для проектирования и сопровождения таких систем, а также средства общения с ними, ориентированные на учителей, школьников и работников аппарата управления органами просвещения.

Учебно-методическое обеспечение школьной информатики состоит в разработке учебных программ, методических пособий, учебников по школьному курсу информатики, а также по всем школьным предметам, которые могут испытывать методологическое влияние информатики, и по курсам, при преподавании которых планируется использование средств информатики. Одна из важнейших задач методического обеспечения — формализация основных понятий и отдельных элементов учебного процесса.

Проблемы организационного обеспечения, связанного с внедрением новых форм как самого школьного учебного процесса, так и управления им, сложны и многообразны. В первую очередь это проблема подготовки и переподготовки преподавательских кадров, способных принести в массовую школу информатику как новую научную дисциплину, как инструмент совершенствования методики других школьных предметов, как стиль мышления.

Хотя учебный процесс в высшей и средней школе имеет некоторые общие черты, специфика средней школы столь существенна, что выделение в самостоятельную прикладную дисциплину характерных задач обработки информации в системе среднего образования представляется целесообразным. Действительно, каждое высшее учебное заведение имеет свой, характерный для него учебный процесс, зависящий от набора узких специальностей этого вуза. Даже среди вузов одного профиля (например, в университетах) особенности учебного процесса в значительной степени зависят от требований региона, от традиций вуза, от планов научной работы его ведущих преподавателей и т. п. В таких условиях вуз гораздо более автономен как система управления по сравнению с типовой средней школой.

Проблемы применения методов информатики к решению основной задачи школы — организации и содержанию ее учебного процесса — могут быть эффективно решены лишь при системном подходе.

Нетрудно видеть, что историческое развитие школьной информатики следует за общим развитием вычислительной техники и ее математического обеспечения, отставая примерно на одно поколение. Это обстоятельство не случайно: школьная информатика как прикладная наука развивается по пути осознания результатов, достигнутых в фундаментальной науке, их

апробирования в других смежных дисциплинах и их оптимальной компиляции для целей системы народного образования. Тем самым школьная информатика в своих прикладных задачах реализует идеи и методы, оптимальность которых доказана предшествующими поколениями вычислительной техники. С одной стороны, этот путь развития соответствует той необходимой степени надежности (иногда ошибочно называемой консерватизмом), с которой обязана работать система народного образования. С другой стороны, этот путь развития соответствует естественному приоритету в расходовании ресурсов общества: технические, финансовые и кадровые средства для перевооружения школы появляются вслед за насыщением производительного сектора средствами вычислительной техники общего назначения и методами их использования.

Образовательное значение школьного курса информатики

Обосновывая необходимость изучения программирования в школе, некоторые авторы полагают, что умение писать программы на тех или иных алгоритмических языках обязательно для каждого, кто по роду своей деятельности должен использовать вычислительную технику. Если бы это было так, то расширение сферы применения ЭВМ автоматически привело бы к необходимости обучения этим языкам в рамках школьного образования.

Однако, понимаемое буквально, сформулированное выше положение ошибочно. По мере распространения вычислительных машин и систем массового пользования вся совокупность взаимодействующих с ними людей все более четко разделяется на две большие группы*:

- системные и прикладные программисты, разрабатывающие системы математического обеспечения и пакеты прикладных программ для решения больших классов задач из самых различных отраслей;
- широкий круг пользователей.

Для представителей первой группы написание программ на тех или иных языках или непосредственно в машинных командах является обязательной составной частью их профессиональной деятельности. Более того, продуктом деятельности системных программистов являются средства общения с ЭВМ, предназначенные для широкого круга пользователей. Совершенствование этого продукта означает в рамках настоящего контекста снижение требований к специальным программистским знаниям и навыкам большинства пользователей.

Общение пользователей с ЭВМ протекает в рамках специализированных, заранее сделанных программных систем, использующих привычную для них профессиональную или бытовую терминологию.

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

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

Рассмотрим наиболее существенные из таких навыков.

Умение планировать структуру действий, необходимых для достижения заданной цели при помощи фиксированного набора средств.

Этот навык часто называют умением алгоритмически мыслить, хотя термин «структура действий» несколько шире классического определения алгоритма.

Деятельность пользователя по описанию алгоритма решаемой задачи состоит в том, что, видя перед собой конечную цель — результат, он конструирует про-

* Обычно выделяют три группы, разделяя системных и прикладных программистов, но с точки зрения рассматриваемых здесь вопросов это различие не имеет принципиального значения.

грамму (в широком смысле этого слова), план действий, представляющий последовательность отдельных более или менее стандартных операций. То, что в обиходе называют иногда программистскими способностями, определяется в первую очередь умением представлять сложное действие в виде организованной совокупности простых.

Важно отметить, что, организуя структуру действий, пользователь должен, как правило, спланировать не только сами действия, но и используемые при этом информационно-технические ресурсы. В каждом таком случае он должен разрешить часто встречающуюся в повседневной жизни диалектическую альтернативу временных и пространственных факторов.

Умение строить информационные структуры для описания объектов и систем.

Человек, использующий накопленные машинными системами информационные фонды, даже будучи освобожденным от описания структур данных, должен тем не менее всегда отдавать себе отчет о классах используемых величин и их взаимосвязях. При этом весьма важен навык представлять себе структуры информационных объектов в формализованном виде.

Значение этого навыка в настоящее время быстро повышается в связи с распространением интегрированных баз данных, информационных систем и АСУ, основу которых составляют информационные модели. При работе с такими моделями необходимо отразить в них *все существенные* для решения поставленной задачи свойства объектов в их взаимодействии, иначе решение, предложенное машиной, сможет сослужить пользователю плохую службу.

Умение организовать поиск информации, необходимой для решения поставленной задачи.

Решение задачи может быть эффективным только в том случае, когда правильно определен объем сведений, необходимых для ее решения, и правильно организован их поиск. При этом вырабатываются навыки формирования поисковых механизмов и пользования ими. Познавательное и методологическое значение этих механизмов выходит далеко за рамки собственно программирования.

По мере роста емкости и снижения стоимости запоминающих устройств ЭВМ становится реальной перспектива передачи машинам всего информационного фонда, накопленного человечеством. Это даст возможность пользователю оперативно получить практически любые необходимые ему сведения. Тем более важным оказывается умение правильно определить, какие именно сведения необходимы и по каким признакам можно организовать их поиск.

Дисциплина и структурированность языковых средств коммуникации, т. е. умение правильно, четко и однозначно сформулировать мысль в понятной собеседнику форме и правильно понять текстовое сообщение.

Отсутствие такой дисциплины в общении людей друг с другом часто компенсируется способностью человека к сопереживанию, позволяющему зачастую правильно понять неправильно высказанную мысль. Машина такой способностью, вообще говоря, не обладает, поэтому всякая неточность в формулировке задания приведет к искажению его смысла и в конечном счете к ошибке.

Существенно, что системный и прикладной программист должен приобрести навык общаться с машиной или системой на различных уровнях в зависимости от их «обученности»: с системами низшего уровня (имеющими незначительное математическое обеспечение) общение возможно только при помощи указаний о «микродействиях» — машинных операциях; с более развитыми системами возникает возможность общения укрупненными предписаниями — процедурами и подпрограммами; в системах с высоким уровнем математического обеспечения программист должен уметь строить свою программу из крупных блоков — операторов языков высокого уровня и готовых программ, в том числе разноязыковых.

Пользователь, далекий от подобного глобального подхода к совокупности решаемых задач, тем не менее обязан, если он хочет рационально общаться с машиной, уметь «запроцедурить» часто используемые им конструкции для того, чтобы впос-

ледствии применять их как элементарные предписания.

Привычка своевременно обращаться к ЭВМ при решении задач из любой области.

Если такая привычка не сформирована, нельзя гарантировать, что даже опытный пользователь, хорошо владеющий перечисленными выше навыками, сообразит обратиться в нужный момент к ЭВМ, если такая задача ему прямо не поставлена. В результате возникают полуанекдотические ситуации, когда программист выполняет вычисления карандашом на бумаге, сидя рядом с терминальным устройством машины, позволяющей сделать это гораздо быстрее и точнее.

Технические навыки взаимодействия с ЭВМ (например, умение работать с клавиатурой).

Разумеется, эти навыки в наибольшей степени определяются уровнем развития техники и практически не влияют на стиль мышления. Более того, не исключено, что развитие систем ввода устной речи в ЭВМ в конечном счете позволит ликвидировать всякую специфику общения с ЭВМ. Тем не менее в обозримой перспективе определенные технические умения необходимы для успешного взаимодействия с ЭВМ.

В философском и педагогическом плане важен тот факт, что значение перечисленных навыков не сводится к обеспечению эффективного взаимодействия с ЭВМ при всей важности этой задачи. Каждый из них играет самостоятельную — и очень важную — роль в той совокупности навыков умственных действий, которая необходима, на наш взгляд, каждому образованному человеку.

Так, умение планировать структуру целенаправленных действий необходимо на любом производстве, в любом научном исследовании, в армии, в быту, в общественной жизни коллектива, в обучении и во многих других сферах.

Информационные модели — это всего лишь частный случай общего понятия **модели**, значение которого для современной науки трудно переоценить. Умение строить информационные модели — частный случай умения правильно строить модели, необходимого сегодня в любом научном исследовании, в любой конструкторской разработке.

Умение организовать поиск нужной информации необходимо при любой научной, исторической, литературной, технической работе, независимо от того, где хранится эта информация — в памяти ЭВМ, в библиотеке, в архиве или патентной картотеке.

Дисциплина общения важна для общения людей между собой отнюдь не меньше, чем для взаимодействия с ЭВМ, и отсутствие этого качества существенно затрудняет разговор. Не менее важно правильно структурировать общение с учетом различного уровня эрудиции и знаний собеседника.

Наконец, умение правильно использовать в нужный момент технические средства — будь то ЭВМ, логарифмическая линейка или карандаш и бумага — само по себе очень важно, и на примере ЭВМ необходимость такого качества только становится более наглядной.

Таким образом, все перечисленные навыки и умения имеют общекультурную, общеобразовательную, общечеловеческую ценность и необходимы в современном мире практически каждому человеку, независимо от его образовательного уровня и сферы приложения его профессиональных интересов. Это определяет необходимость их формирования на этапе *школьного образования*.

Делаются попытки обсудить пути формирования алгоритмической культуры школьника при обучении математике (например: *Монахов В. М.* и др. Формирование алгоритмической культуры школьника при обучении математике: Пособие для учителей. М., 1978). Авторы используют некоторые из фундаментальных понятий программирования и даже терминологию и символику языков программирования.

По-видимому, это не случайно. На наш взгляд, такая задача и не может быть решена в рамках существующих учебных дисциплин, так как ни один из школьных предметов, более того — ни одна из наук, содержание которых в них отражено, не обладает достаточно развитым концептуальным запасом для выполнения соответствующих действий.

В то же время программирование, или,

точнее говоря, информатика, таким инструментарием располагает.

В самом деле, для **планирования структуры действий** предназначен весьма мощный аппарат средств управления в языках программирования: разветвлений, циклов процедур, рекурсивных вызовов, макроопределений, средств синхронизации параллельных процессов (семафоров Дейкстры и т. д.), не говоря уже о сложных способах и принципах планирования, используемых в операционных системах.

Для **информационного моделирования** объектов и систем служат структуры данных, начиная от простых переменных и кончая развитыми системами — иерархическими, сетевыми, реляционными, процедурно-дедуктивными.

Аналогично для **структурирования процесса общения** служат синтаксические описания и макросредства, для **организации информационного поиска** — различные поисковые механизмы (каталоги, многоуровневые индексы, функции расстановки и т. д.) и средства доступа в системах управления базами данных.

Поэтому, на наш взгляд, КУРС ИНФОРМАТИКИ предназначен для формирования описанного выше стиля мышления.

Слово «курс» мы используем условно, понимая под этим всю совокупность идей, методов, понятий и средств информатики, включаемых в программу общеобразовательной школы, вместе с необходимым для преподавания учебно-методическим обеспечением, независимо от того, изучаются ли они в рамках информатики как отдельного предмета или внутри других школьных дисциплин.

Место информатики в общеобразовательной школе

Выше уже подчеркивалось большое общекультурное, методологическое значение основных «программистских» навыков мышления. В этом отношении они могут быть поставлены в один ряд с развитием количественных и пространственных представлений, с умением *абстрагировать*, *схематизировать* и с другими элементами математического развития.

Поэтому формирование этих навыков должно начинаться одновременно с выработкой основных математических понятий и представлений, т. е. в *младших классах общеобразовательной школы*. Только при этом условии программистский стиль мышления сможет органично войти в систему научных знаний, навыков и умений, формируемую школой. В более позднем возрасте формирование такого стиля может оказаться связанным с ломкой случайно сложившихся привычек и представлений, что существенно осложнит и замедлит этот процесс.

Раннее изучение информатики целесообразно и по другой причине: это даст возможность при изучении других учебных дисциплин вырабатывать упоминавшуюся выше привычку своевременного обращения к ЭВМ непосредственно в ходе освоения соответствующих разделов программы, опираясь на имеющиеся навыки взаимодействия с ЭВМ.

Все содержание общеобразовательного курса информатики можно разделить на четыре относительно самостоятельные части.

Это, во-первых, совокупность наиболее фундаментальных навыков, знаний, понятий и представлений, необходимых для формирования *программистского стиля мышления*.

Во-вторых, это совокупность прикладных навыков и умений, необходимых для применения идей и методов информатики в других отраслях человеческой деятельности.

В-третьих, это система основных положений информатики как науки в соответствии с ее местом в современной системе научных знаний.

И в-четвертых, это комплекс знаний, необходимых для общей ориентации в возможностях современной и перспективной вычислительной технике прикладных знаний о стандартных языках, информационных системах, о некоторых физических и логических принципах работы ЭВМ и т. д.

Фундаментальные знания и навыки первой группы, как указывалось выше, могут и должны быть сформированы в младших классах.

Прикладные навыки второй группы

должны в основном формироваться при изучении других предметов школьного курса — от математики до литературы — по мере освоения соответствующих фундаментальных понятий.

Наконец, систематизация полученных фундаментальных и прикладных знаний и ознакомление с общим уровнем развития вычислительной техники должны осуществляться в конце школьного образования, т. е. в старших классах средней школы.

Учитывая это разделение, можно предложить следующую схему преподавания информатики в школе.

II—V классы	IV—VIII классы	IX класс	X класс
Совокупность наиболее фундаментальных навыков, знаний, понятий и представлений, необходимых для формирования программистского стиля мышления	Совокупность прикладных навыков и умений, необходимых для применения идей и методов информатики в других отраслях человеческой деятельности	Система основных положений информатики как науки в соответствии с ее местом в современной системе научных знаний	Комплекс знаний, необходимых для общей ориентации в возможностях современной и перспективной вычислительной техники, прикладных знаний о стандартных языках и системах

В младших классах (со II по V) информатика преподается как предмет, основное содержание которого составляют знания и навыки первой из перечисленных групп. Специфика освоения этих навыков, а также традиции начальной школы, в силу которых все предметы ведет один учитель, делают целесообразным постановку в начальной школе единого курса «математика—информатика—язык (русский язык)».

Освоение навыков второй группы происходит в рамках других предметов в течение всего срока обучения в школе. При этом нет необходимости выделять информатику в отдельный предмет — весь программистский инструментарий к этому времени уже, по существу, сформирован. Применения его погружаются в учебные программы других предметов, главным образом в центральных классах средней школы, с IV по VIII.

Первыми среди школьных предметов, которым призвана служить информатика, следует назвать математические дисциплины.

Информатика не призвана ни заменить собою ни один из школьных предметов, ни изменить роль предмета в общей системе школьных дисциплин, ни перераспределить соотношение гуманитарных и естественных дисциплин в школе. Она лишь предлагает каждой из дисциплин, изучаемых в школе, новый и весьма совершенный инструмент, который позволяет учителю, умеющему пользоваться этим инструментом, глубже и эффективнее раскрыть перед школьниками сущность своего предмета. При этом нельзя назвать ни одного школьного предмета, в котором аппарат информатики оказался бы бесполезным.

Из этих замечаний нетрудно сделать вывод: школьный курс информатики является не дополнительной нагрузкой на школьника, а важнейшим средством уменьшения его перегрузок, сокращения и уплотнения программы средней школы в целом. Не следует забывать, конечно, что кроме такого «инструментального» значения школьный курс информатики всегда сохраняет *собственную философскую значимость*.

Наконец, в старших классах в школе появляется самостоятельный предмет — информатика. Его задачи — подвести итог многолетнему знакомству с информатикой, систематизировать полученные знания (IX класс), ввести в курс современного состояния и перспектив развития вычислительной техники (X класс). В то же время отдельные прикладные знания о принципах работы ЭВМ могут войти в учебные программы других предметов (физика, математика и др.).

Надо отметить, что всякая школьная дисциплина наряду с обязательным курсом обеспечивает возможность организации

внеклассной и внешкольной работы, позволяющей закрепить и расширить полученные знания и навыки, осуществить профессиональную ориентацию, допрофессиональную подготовку и трудовое воспитание учащихся.

В этом отношении у информатики особенно богатые возможности. Школьники, усвоившие основные положения фундаментального курса, могут сразу же перейти к освоению основных элементов системного и прикладного программирования и стандартных языков и систем, что позволяет им уже в IV—V классах участвовать в разработке реального математического обеспечения в составе школьных производственных коллективов. Следует подчеркнуть, что в таких коллективах учащиеся привлекаются к производительному труду в его наиболее увлекательных и современных формах, что особенно важно для осуществления трудового воспитания. Другая форма внеклассной работы — летние школы юных программистов.

О подходе к преподаванию основ программирования

Выбирая стратегию преподавания информатики в школе, необходимо учитывать, что задача общеобразовательного курса — это в большей степени выработка определенного стиля мышления, формирование наиболее общих навыков, умений и представлений, нежели освоение тех или иных конкретных языков и технических средств программирования.

В то же время такой курс должен служить базой для последующего профессионального изучения программирования в высшей школе, в старших классах средней школы (в рамках профессионального обучения); во внешкольных детских учреждениях, ученических производственных коллективах и т. д.

В настоящее время существует три наиболее распространенных подхода к преподаванию программирования:

- изучение одного или нескольких языков программирования, широко используемых при решении научных и хозяйственных задач (в дальнейшем мы будем называть такие языки *стандартными*);
- преподавание программирования как теоретической дисциплины вообще, без освоения конкретных языков и систем;
- преподавание на основе специально разработанного языка, ориентированного на обучение основным навыкам программирования.

Первый подход, вероятно, во многом оправдан при обучении программированию в рамках профессиональной подготовки (во всяком случае он является обязательной составной частью такой подготовки). Однако для общеобразовательной школы такой подход непригоден прежде всего потому, что ни один из существующих стандартных языков не отражает в достаточно чистом виде современный концептуальный запас программирования. Такое положение, вообще говоря, естественно, так как все эти языки разрабатывались с совершенно другими целями и каждый из них ориентирован на определенную, более или менее узкую область применения.

Кроме того, большинство реализаций стандартных языков загружено большим количеством технических деталей, связанных с особенностями транслятора или объектной машины, но совершенно не нужных для усвоения основных приемов мышления.

Второй подход представляется на первый взгляд наиболее логичным. Однако его практическая реализация наталкивается на серьезные трудности, так как с отказом от языка программирования не только теряется возможность использовать соответствующий инструментарий, но и становится зачастую трудно обосновать необходимость его использования.

Широко известна, например, методика обучения, принятая в Симферопольской школе юных кибернетиков. В соответствии с этой методикой алгоритмический стиль мышления вырабатывается в ходе изучения основ теории нормальных алгоритмов Маркова, а также абстрактных машин Тьюринга и Поста. К сожалению, этот подход позволяет освоить лишь одну сторону такого стиля — умение пользоваться ограниченными средствами так как инструмен-

тарий всех перечисленных систем крайне беден и включает, по сути, только разветвление и переход, что удобно для теоретических исследований, но не для практики.

Другой пример, в некотором смысле крайний по отношению к предыдущему, но в то же время и аналогичный, прямое обучение навыкам планирования при помощи составления блок-схем, например, для планирования собственной деятельности. Наряду с концептуальной ограниченностью такая методика имеет и другой недостаток: отказ от применения реальной или хотя бы гипотетической машины в качестве объекта планирования деятельности не позволяет убедительно для учащегося мотивировать самую необходимость такого планирования. Действительно, если для ЭВМ выполнение какого бы то ни было задания без программы просто невозможно, то для человека предварительный план часто можно заменить «ориентированием по обстановке».

Важно отметить, что попытки построения курсов программирования исключительно на базе использования блок-схем, с одной стороны, вынуждены ориентироваться на такие бедные наборы механизмов программирования, которые включают лишь бинарные ветвления и переходы, а с другой стороны, даже на неветвящихся участках блок-схемы, внутри «блоков», неявно использовать описания данных и операций языковыми средствами. В «блок-схемном» изложении программирования уходит на второй план такая важная черта современного программирования, как структурирование, которая, кстати, обеспечивает наглядность программ, не прибегая к рисованию.

Таким образом, единственно приемлемым для общеобразовательной школы представляется третий подход — обучение на базе специально созданного языка, отражающего все основные концепции современного программирования. В последнее время даже в высшей школе такой подход находит все больше сторонников — именно с таких позиций создавались языки ПАСКАЛЬ и SP/k, предназначенные для обучения студентов.

Интересно отметить, что Бэйсик, создававшийся как учебный язык, получил широкое распространение в качестве стандартного, но его педагогическая ценность оспаривается многими авторами. Вероятно, дело в том, что его разработчики поставили своей целью создать максимально простую в изучении систему, не стараясь в то же время отразить в ее структуре какую-либо цельную систему понятий и средств. В результате освоить этот язык действительно легко, но концептуально язык оказался крайне бедным, так что перейти после него к какому-либо другому языку или просто научиться программировать в хорошем стиле очень тяжело.

Известны и специализированные языки для младших школьников. В их числе можно назвать Школьник (СССР), SMP (Великобритания), LOGO (США). Первые два языка предельно упрощены, их можно рассматривать лишь как робкие попытки оценить возможности младших школьников в области программирования. Большой интерес представляет язык LOGO, разработанный С. Пайпертом в Массачусетском технологическом институте, но его эффективное использование возможно лишь в обстановке терминального класса, когда каждый учащийся имеет отдельный терминал для связи с ЭВМ.

Требования к языку начального обучения программированию

Основные требования к языку начального обучения программированию для общеобразовательной школы, вытекающие из перечисленных выше основных положений, можно сформулировать в следующем виде:

1. Язык и связанная с ним система программирования должны отражать в своей структуре *все основные концепции программирования*.

2. Язык должен быть логически ясным и методически обусловленным, т. е. его структура должна быть подчинена определенной методической схеме, позволяющей последовательно вводить новые понятия, формировать перечисленные выше навыки и умения.

3. Язык не должен содержать конструкций, определяемых исключительно логикой транслятора или объектной машины

(и тем более противоречивых конструкций).

4. Язык должен иметь русскую лексику и допускать возможность создания версий на языках других народов СССР.

5. Синтаксис может быть близким к синтаксису русского языка, но не должен совпадать с ним, чтобы была возможность подчеркнуть специфику общения с ЭВМ. В то же время семантика операторов не должна противоречить семантике соответствующих предложений естественного языка.

6. Структура учебного языка должна *методически сопрягаться* со структурой современных стандартных языков программирования высокого уровня, т. е. изучение этого языка должно облегчать или по крайней мере не затруднять последующий переход к изучению других языков, например, в рамках профессиональной подготовки. Из этого положения следует, что, во-первых, языковые конструкции учебного языка должны быть близкими по структуре к соответствующим конструкциям стандартных языков, и во-вторых, что некоторые возможности могут быть даже сознательно исключены из языка, чтобы создать мотив для последующего перехода к более сложным стандартным языкам.

7. Язык и система программирования должны быть достаточно мощными, чтобы обеспечить возможность их применения при изучении других дисциплин.

8. Язык должен допускать возможность использования различных объектных машин. Более того, должна быть предусмотрена возможность освоения этого языка вообще без выхода на реальные ЭВМ. Поэтому синтаксис языка должен допускать написание программ для гипотетических машин, в том числе роботов, самоходных аппаратов и т. д., действия которых школьники могли бы легко представить.

9. Желательно, чтобы система программирования обеспечивала сопряжение с автоматизированными обучающими и контролирующими системами (в частности, транслятор должен допускать организацию сбора статистики об ошибках).

10. Транслятор должен обеспечивать развитую диагностику на естественном языке.

Сформулированные положения (в частности 1, 6 и 7) с необходимостью требуют, чтобы язык начального обучения был структурированным, содержал хорошо развитый аппарат ветвлений, циклов, процедур, достаточно богатые структуры данных, располагал средствами отладки и документирования программ.

Для реализации первого из перечисленных требований необходимо построить методически обоснованную *систему основных понятий* и представлений, которые должны быть усвоены школьниками в рамках общеобразовательного курса информатики и, следовательно, должны быть отражены в конструкциях языка.

На наш взгляд, такая система должна обязательно включать следующие элементы:

Программа — задание для ЭВМ и роботов, составленное в принятой для них форме.

Предписание — часть программы, представляющая собой предложение, которое может быть исполнено.

Система предписаний — множество предписаний, которые могут быть употреблены в программе для определенного робота, машины или системы.

Память — устройство для хранения программы и данных.

Процедура — программа или самостоятельная часть ее, которая может храниться в памяти и вызываться по имени.

Переменное поле — часть предложения, которая может быть заменена любым элементом некоторого множества (множества значений).

Имя — определенным образом обозначенное переменное поле в тексте программы, которое принимает значения в процессе ее исполнения.

Условное предписание — предписание, характер исполнения которого зависит от истинности или ложности некоторого утверждения.

Цикл — многократное выполнение некоторого предписания.

Внутреннее имя (параметр) — имя в теле процедуры, неизвестное за ее пределами и определяющее, вообще говоря, характер выполнения процедуры.

Функция — процедура, вырабатываю-

щая значение (результат), зависящее, вообще говоря, от значений параметров.

Данные — совокупность значений, встречающихся в программе имен.

Структуры данных — совокупности данных, описывающие реальные объекты или системы.

На основе этой системы понятий строится полная методическая схема курса, включающая «методические связки» и вспомогательные понятия, необходимые для эффективного усвоения школьного курса информатики.

Легко видеть, что предлагаемый курс практически не требует предварительных знаний, выходящих за рамки программы I класса.

Этапность внедрения школьного курса информатики

Естественный путь внедрения школьной информатики представляется эволюционным, включающим несколько этапов, уровней, которые соответствуют различной степени технической оснащенности школ и подготовленности кадров.

Высший уровень технической оснащенности предполагает наличие терминала (дисплея) на каждом школьном рабочем месте. Ученик имеет возможность постоянно и оперативно общаться с доступными информационными, справочными, контролирующими обучающими системами.

Более близкая перспектива системы народного образования связывается с переходным периодом от типовой сегодняшней школы к школе будущего. В школе переходного периода установлены 2—3 терминала. Школьники имеют возможность общения с машиной в порядке лабораторных работ, которые при рациональной организации учебного процесса могут охватывать несколько школьных дисциплин. Учебная программа по курсу информатики учитывает это ограничение технического обеспечения. К числу школ переходного периода можно отнести небольшое число экспериментальных школ и внешкольных учреждений, имеющих контакты с крупными ВЦ в важнейших научных центрах.

Непосредственно начальный этап работ по постановке школьного курса информатики относится к типовым сегодняшним школам. Впрочем, и среди них выделяются школы, расположенные в крупных городах, научных, промышленных и сельскохозяйственных центрах, имеющих достаточные ресурсы информационных и вычислительных мощностей и обеспечивающих по крайней мере эпизодические контакты школьников с ЭВМ. Как правило, такие школы имеют факультативы и кружки, руководимые профессиональными программистами.

Несомненно, однако, что предмет главной заботы — все общеобразовательные школы страны, включая отдаленные сельские и малокомплектные, массовые школы. Возможность создания в таких школах технической базы для преподавания информатики зависит от общих темпов развития информационных сетей, вычислительных центров коллективного пользования (ВЦКП) и средств связи. Вероятнее всего, в обозримой перспективе трудно рассчитывать на установку терминального оборудования во всех, без исключения, школах. В то же время преподавание основ информатики с эпизодическим выходом на ЭВМ (в районных центрах) может быть организовано практически во всех школах страны по мере решения проблемы подготовки преподавательских кадров.

Уже в сегодняшней массовой школе многие важнейшие понятия и навыки, связанные с программированием, могут быть сформированы путем использования возможностей заочных вычислительных центров, хотя, разумеется, не в полном объеме и менее глубоко.

Система математического обеспечения (СМО), ориентированная на школьный учебный процесс

Одна из задач математического обеспечения, ориентированного на учебный процесс, — приблизить операционную обстановку обучения к реальной операционной обстановке, в которой окажется школьник после окончания обучения. Для решения этой задачи необходимо организовать

структуру школьного математического обеспечения таким образом, чтобы в ней были отражены основные формы взаимодействия с ЭВМ в научной, производственной, бытовой и других сферах.

Общая схема отражения, основанная на современных представлениях о перспективах развития и применения вычислительной техники, изображена на рис. 1. В соответствии с этой схемой, основные компоненты, вносимые вычислительной техникой в производственную операционную обстановку, — интегральные банки данных, пакеты прикладных программ, микропроцессорные средства управления — отображаются в соответствующих элементах школьной операционной обстановки.

Построенная в соответствии с этими требованиями общая структура СМО школьного учебного процесса представлена в виде многослойной круговой диаграммы на рис. 2. В верхнем секторе диаграммы указаны пользователи, на которых ориентированы соответствующие программные слои. В остальных секторах перечислены компоненты системы.

Краткий обзор основных элементов системы удобнее всего начать с третьего слоя, который можно назвать инструментальным комплексом школьника. Важнейшая компонента этого слоя — **язык начального обучения** и связанная с ним система программирования (транслятор, система диагностики и сбора статистики, макрогенератор и т. д.). Основное назначение учебного языка — способствовать формированию у школьников программистского стиля мышления, помочь им в ус-

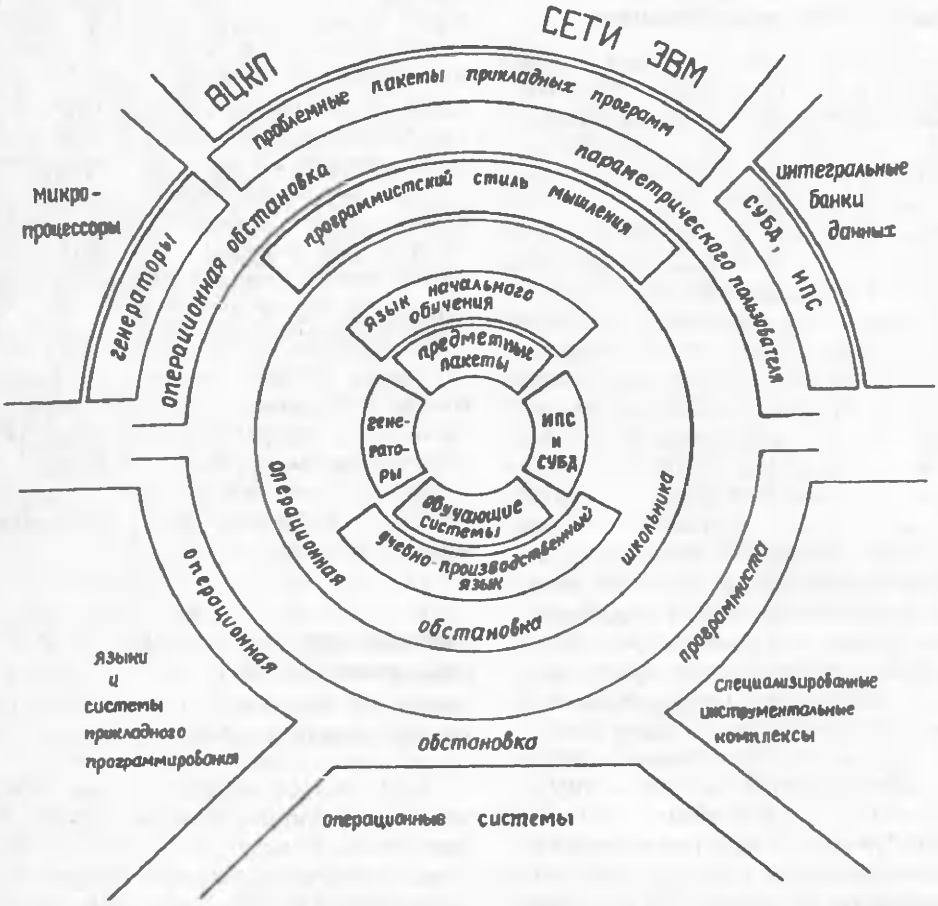


Рис. 1

воении основных понятий и навыков умственных действий, необходимых для эффективного использования вычислительной техники и имеющих, кроме того, большую общеобразовательную ценность.

В ходе решения задач, возникающих в школьной практике, учащимися может быть накоплен **архив прикладных программ** на языке начального обучения, организованный в виде групповых и личных библиотек и относящийся к первому, внешнему программному слою.

Второй элемент школьного инструментального слоя — **учебно-производственный язык** (вместе с соответствующей системой программирования). Это универсальный язык высокого уровня, позволяющий средним и старшим школьникам (VI—X классы) создавать эффективные программы для решения учебных и производственных задач, закрепляя тем самым навыки и умения, полученные в ходе начального обучения, и в то же время позволяя использовать возможности вычислительной техники для решения практических задач по другим школьным предметам. Такой язык должен иметь достаточно универсальный и мощный набор конструкций, близкий к современным языкам высокого уровня (ПАСКАЛЬ, АЛГОЛ-68, ПЛ/1), и допускать эффективную реализацию на современных ЭВМ. Легко видеть, что эти свойства противоречат сформулированным выше требованиям к языку начального обучения, поэтому использовать один и тот же язык и в качестве начального, и в качестве учебно-производственного не представляется целесообразным.

Рис. 2

В то же время требования логической ясности, стройности, отсутствия громоздких конструкций, удобства структурирования, отладки и документирования сохраняют силу и для учебно-производственного языка. Кроме того, такой язык должен допускать организацию макрорасширений, ориентированных на определенный класс задач по одной из школьных дисциплин. По-видимому, ни одна из производственных систем программирования, широко эксплуатируемых сегодня, не удовлетворяет этим требованиям в полном объеме, так что разработка самостоятельного учебно-производственного языка для школьников остается актуальной задачей.

Реализацией требований к учебно-производственному языку является система программирования Рапира, построенная на базе адаптированного к условиям обучения и работы школьников объединения полезных возможностей Сетла и Поплана (диалог, богатый набор структур данных, динамические типы, конструирование функций и операций).

Так же как и в случае языка начального обучения, прикладные программы на учебно-производственном языке образуют архив, относящийся к внешнему программному слою.

Третий элемент — **система управления базами данных** с соответствующими языковыми средствами, позволяющими школьникам формировать индивидуальные и групповые базы данных и использовать их в учебе, в быту и в других областях деятельности, одновременно приобретая навыки создания и использования программно независимых информационных структур. Подготовленные школьниками **ученические базы данных** относятся к внешнему слою программно-информационного обеспечения вместе с библиотеками прикладных программ на учебно-производственном и начальном языках.

Второй слой составляют программы и программные системы, для которых школьники являются только пользователями. Разумеется, такая классификация в достаточной степени условна, в частности, важнейший элемент второго слоя — предметные пакеты — захватывают частично и первый, и третий слой, как это видно из рисунка.

Предметные пакеты — это небольшие, как правило, пакеты прикладных программ, предназначенные для решения определенных, достаточно узких классов задач по тематике школьных учебных предметов и в смежных областях деятельности. Входной язык такого пакета должен в максимально возможной степени соответствовать терминологии и системам обозначений, принятым в соответствующей области.

Пожалуй, наиболее интересный класс предметных пакетов — моделирующие программы, позволяющие воспроизвести средствами вычислительной техники ход того или иного физического, химического, биологического, технического эксперимента. Легко можно представить себе, например, предметный пакет по неорганической химии, позволяющий моделировать протекание и результаты реакций, записанных в обычной химической символике. Аналогичные пакеты могут быть созданы для ядерных реакций, синтеза органических соединений, биохимических процессов, электро- и радиосхем, механических, аэро-, гидро- и астродинамических процессов и т. д.

Создание таких пакетов позволит многократно усилить лабораторную базу каждого учебного предмета и в то же время приучить школьников к формам использования вычислительной техники, характерным для соответствующей дисциплины. Наличие достаточного числа моделирующих пакетов позволит в конечном счете реализовать принцип «машины-лаборатории», сформулированный У. Аттелем.

Другой класс предметных пакетов строится на основе машинной реализации имитационных игр, которые могут использоваться во всех школьных дисциплинах, изучающих сложные динамические (эволюционные или революционные) процессы в природе и обществе. Такие игры создаются или могут быть созданы для уроков истории, биологии, географии (геологии, экономики), астрономии. В имитационной игре в ходе урока с помощью ЭВМ ученик может активно воспроизвести историко-политическую, геологическую и т. п. ситуацию, глубже и прочнее усвоить причинно-

следственные отношения между эволюционными процессами и явлениями. В предметный пакет такого класса входит не только программная реализация игры, но и программное обеспечение анализа ее результатов.

Третий класс предметных пакетов — операционные, позволяющие реализовать средствами вычислительной техники наборы операций, характерных для той или иной дисциплины. Примерами таких пакетов могут служить системы для аналитических преобразований, пакеты средств машинной графики, системы для верификации программ и автоматического доказательства теорем и т. д.

Некоторые операционные пакеты занимают промежуточное положение между вторым и третьим программными слоями. Например, в ближайшие годы могут получить развитие генераторы моделей микропроцессоров с заданными свойствами. Сами эти модели могут в свою очередь использоваться в учебных целях, образуя тем самым внешний программный слой. Наконец, отдельные предметные пакеты могут создаваться самими школьниками с помощью учебно-производственного языка или даже с помощью языка начального обучения. Примером может служить эффективно используемый в учебном процессе пакет графических средств ШПАГА. Такие пакеты также можно отнести к внешнему программному слою.

К предметным пакетам примыкают **базы данных по учебным дисциплинам** и связанные с ними справочные и информационно-поисковые системы с соответствующими языками запросов. В отличие от ученических баз данных, описанных выше, информация в учебных БД организуется преподавателями (в более отдаленной перспективе — органами народного образования в централизованном порядке), а соответствующие языковые средства создаются системными программистами (совместно с преподавателями). Очевидно, что средства доступа для учебных и ученических баз данных должны быть максимально унифицированы.

Наконец, ко второму программному слою относятся также обучающие и контролирующие программы. Это, пожалуй, единственный элемент «школьных» слоев математического обеспечения, для которого накоплен довольно значительный опыт, в том числе в отечественной практике. В то же время обучающие программы могут быть эффективны в общеобразовательной школе только при наличии достаточно больших ресурсов терминального времени (несколько часов в неделю в расчете на одного ученика) и при наличии достаточно обширных централизованных банков данных общего пользования, поэтому в ближайшие годы практическое применение большого количества таких программ в широких масштабах представляется маловероятным.

Остальные элементы «школьного» слоя допускают побригадную эксплуатацию, и их использование вполне можно начинать при наличии одного-двух терминалов на всю школу.

Четвертый программный слой — это прежде всего инструментальный комплекс учителя, позволяющий квалифицированному преподавателю создавать или корректировать компоненты школьных слоев. В состав такого комплекса входят: генератор предметных пакетов, предназначенный для генерации новых и корректировки имеющихся пакетов по учебным дисциплинам, система управления учебными базами данных, генератор «роботов» для языка начального обучения, позволяющий преподавателю создавать системы команд гипотетических машин и устройств, макросредства для расширения и корректировки учебно-производственного языка и генератор обучающих курсов.

Помимо инструментального комплекса к четвертому слою можно отнести разнообразные программы, для которых школьный преподаватель является только пользователем, — прежде всего системы АСУ школой, включающие в себя программы управления хозяйственной и учебной деятельностью школы, сбора учебной статистики и т. д. Эти системы в основном аналогичны по своим задачам существующим программам АСУ—вуз, поэтому проблемы, возникающие при их разработке, хорошо известны.

Все основные компоненты третьего-четвертого слоев и некоторые программы второго слоя должны создаваться систем-

ными программистами. При этом необходимо учесть как специфику пользователей, так и особенности реализации системы.

Поэтому в составе специализированного математического обеспечения, ориентированного на школьный учебный процесс, предусмотрен пятый слой — инструментальный комплекс программиста, позволяющий создавать математическое обеспечение.

Наконец, ядро любой системы математического обеспечения составляет операционная система (ОС) объектной ЭВМ. Точнее, речь идет о двух взаимосвязанных системах: ОС периферийного центра обработки и ОС базового вычислительного комплекса. Вообще говоря, ядро является сменным и меняется при переходе от одной модели ЭВМ к другой. Однако эти изменения должны отражаться только на самом внутреннем, пятом программном слое, который тем самым «экранирует» остальные слои и обеспечивает переносимость системы математического обеспечения в целом при смене моделей и поколений ЭВМ.

Учебно-методическое обеспечение

Современная информатика рассматривает ЭВМ как универсальное средство для обработки любого вида представимой в машине информации — текстовой, управляющей и, только в частном случае (далеко не самом важном, не самом распространенном), цифровой. Обеспечить глубокое понимание идей, методов и фундаментальных понятий можно, лишь отказавшись с первых шагов изучения программирования от доминирующего влияния арифметизации.

Учебник программирования для школьников II—V классов должен отвечать всем перечисленным выше требованиям к языку и в то же время иметь простой стиль изложения, увлекательные иллюстрации, тщательно отработанный методический аппарат. Наряду с учебником необходимо подготовить задачник по курсу с разнообразными задачами, ответами-программами, методическими указаниями для преподавателя.

Перспективы компьютеризации системы образования

Слово *компьютеризация* в применении к любой сфере деятельности означает в первую очередь внедрение автоматизированной системы управления. Вообще говоря, закономерен вопрос о проектировании и внедрении АСУ школой. Однако при всей важности этого вопроса, заслуживающего самого пристального изучения в органах управления образованием, его нельзя считать неисследованным, принципиально новым, во всяком случае с точки зрения информационного и программного обеспечения.

Проектирование АСУ школой, включающей в себя генерацию школьной отчетности органам народного образования, выдачу внутришкольной документации по посещаемости и успеваемости, статистический анализ учебной работы в классах, т. е. рутинная деятельность, даже составление расписания, сводится в конечном счете к использованию опыта и рекомендаций проектировщиков многочисленных типовых АСУ.

В настоящее время уже разработаны и действуют отдельные подсистемы и программы школьных АСУ, в реализации которых участвовали и сами школьники. Развитие вычислительной техники и ее математического обеспечения, безусловно, оказывает влияние на АСУ школой, но, по-видимому, и в будущем техническое и математическое обеспечение школьных АСУ будет непосредственно следовать за общим развитием вычислительной техники.

Когда речь идет о школе, в слово *компьютеризация* вкладывается и более широкий смысл: автоматизированное управление основным видом деятельности — учебным процессом, не только его формой, но и содержанием. Все возможности автоматизации учебного процесса должны самым внимательным образом изучаться в связи с реальной перспективой значительного усложнения проблем, стоящих перед образованием: недостаток квалифицированных педагогических кадров в массовой школе; увеличение и усложнение информационных потоков в обществе, требующих непрерывного усложнения школьных про-

грамм; «ножницы» массовости образования и требований индивидуального подхода к учащимся.

Попытки построить алгоритмы учебного процесса и реализовать их с помощью ЭВМ были сделаны еще в 60-е гг., когда весь мир был захвачен волною увлечения программированным обучением. В это время было выполнено много интересных работ, которые в целом показали принципиальную возможность обучения с помощью ЭВМ учащихся самых разных возрастов различным дисциплинам.

Тогда же были построены разнообразные автоматизированные системы контроля знаний, автоматические экзаменаторы и т. д. Удалось не только реализовать содержательный и разветвленный диалог машины-преподавателя и школьника, но и оптимистически оценить психологические аспекты такого диалога. В частности, было показано, что программированное обучение — это один из путей для определения оптимальной, смешанной стратегии между принципами массовой школы и индивидуального подхода к способным ученикам.

Одним из главных методологических факторов компьютеризации школы является весь арсенал существующих и развиваемых средств управления большими базами данных. В условиях сетей и центров коллективного пользования появляется возможность автоматизировать не отдельный урок или отдельный курс, а создавать единую базу данных образования, которая будет содержать многовариантные материалы уроков и курсов, методические планы, учебные программы и т. п. Важно, что база данных образования включает не только наборы понятий, но и их зависимости, содержательные и методические.

Очевидна необходимость и централизованного фонда образования, и локальных баз данных различных уровней (республика, область, район, школа). Таким образом, информационные фонды образования представляются в виде системы распределенных баз данных с многоуровневым коллективным использованием. В таких условиях можно оперативно контролировать, обсуждать, распространять, оптимизировать все виды документации в системе образования: программы курсов, методики, планы уроков и т. п.

Фундаментальным инструментом компьютеризации учебного процесса представляются дедуктивные информационные системы, которые в состоянии осуществлять логический анализ сообщаемых машине сведений с целью вывода новых фактов или проверки их непротиворечивости с имеющимися в системе сведениями.

Важное для школы преимущество таких систем заключается еще и в том, что в результате автоматического заполнения и сложного логического анализа больших баз данных могут быть получены нетривиальные следствия, вытекающие из разнородных, внешне не связанных между собой фактов. Такие системы при эффективной их эксплуатации могут стать, по образному выражению академика В. М. Глушкова, «коллективным мозгом».

Внушают оптимизм разработки средств автоматического распознавания семантики текста. Возможность общения с машинами на естественном языке при работе с универсальными базами данных призвана освободить пользователей автоматизированной системы образования от большинства технических трудностей общения с машиной.

Легко видеть, что в программном обеспечении школьных вычислительных систем средства программированного обучения оказываются лишь одной из многочисленных компонент. В этом свете кажутся малоперспективными автономные вычислительные системы, узко ориентированные только на задачи программированного обучения. Это обстоятельство не снижает, конечно, значимости накопленного опыта, средств и методов программированного преподавания с помощью ЭВМ ни сегодня, ни в перспективе, если речь идет о решении отдельных задач или вспомогательных подсистем школьного учебного процесса.

В заключение приведем слова академика Г. И. Марчука: «...проблема обучения программированию и проблема обучения *вообще с помощью ЭВМ*... является стержнем интенсивного прогресса в области образования, во многих областях науки».

Фирма АЛТ – НТП «Альтернатива» – НТП «Новая Альтернатива»

ИСТОРИЯ

1986 г. В школах Алтайского края внедряется курс информатики. В Алтайском ИУУ (г. Барнаул) создается кабинет информатики. Его возглавляет А. Н. Гриценко, до этого работавший на кафедре прикладной математики АлтПИ и окончивший в 1985 г. аспирантуру математического факультета Новосибирского университета. Нет ни техники, ни программного обеспечения, ни методики. Во всем крае имеется только три терминальных класса КУВТ-86.

1987 г. Количество терминальных классов в Алтайском крае достигает шести. Попытки найти для них программное обеспечение в Москве и других городах не имеют успеха. Во всех вузах и лучших школах установлены компьютеры «Ямаха» и IBM, а для массовой школьной техники практически ничего нет. Немногие разработки такого качества, что для учебного процесса по ОИВТ непригодны. На одном из совещаний А. Н. Гриценко в ответ на свое резкое заявление о низком качестве одной из таких систем слышит от директора ИУУ: «А у вас есть альтернатива?»
В течение последующих трех месяцев в кабинете информатики Алтайского ИУУ создается первый программно-методический пакет для классов КУВТ-86, получивший название «АЛЬТЕРНАТИВА». Пакет замышлялся как комплект программ, документации и методик, который ИУУ должен передать учителю ОИВТ. Никаких задач, кроме обеспечения нормальной работы имеющихся в крае шести терминальных классов, не ставилось.

1988 г., февраль Неожиданно для разработчиков созданный пакет «АЛЬТЕРНАТИВА» приобретает популярность. Поступают многочисленные запросы от школ и учителей других областей. Становится ясна необходимость перевода разработок на постоянную и хозрасчетную основу. При кабинете информатики ИУУ была создана кооперативная фирма АЛТ. Руководитель — А. Н. Гриценко. Программный пакет фирмы быстро совершенствуется и приобретает широкую известность, на него ежедневно поступают заказы из всех регионов и республик СССР. Большинство заказчиков — небогатые сельские школы.

1990 г. В Алтайском крае устанавливаются десятки новых классов типа «Корвет» и УКНЦ. Поступают многочисленные просьбы сделать аналог пакета «АЛЬТЕРНАТИВА» для этих классов. Работа по созданию этих пакетов потребовала полного перехода на профессиональную основу. Создается совместное с Институтом информатики АПН малое предприятие «Альтернатива». Его сотрудниками становятся работники Алтайского ИУУ.

1991 г. Предприятие «Альтернатива» разрабатывает первые версии программно-методических пакетов «АЛЬТЕРНАТИВА-КОРВЕТ» и «АЛЬТЕРНАТИВА-УКНЦ». Начинаются их поставки многочисленным заказчикам. В предприятии начинаются внутренние разногласия. Недовольный диктатом большинства и снижением темпов и качества работ основатель и руководитель предприятия А. Н. Гриценко выходит из него и основывает обновленное предприятие «Новая Альтернатива».

1992–1993 гг. НТП «Новая Альтернатива», начав с нуля, быстро создает новые версии пакетов «АЛЬТЕРНАТИВА» для классов УКНЦ, «Корвет» и КУВТ. Для классов КУВТ-86 создаются пакеты «РОБОТЛАНДИЯ» и «ЛОГО». Устанавливаются агрессивно низкие цены на пакеты, доступные практически всем заказчикам.

1994 г. НТП «Новая Альтернатива» создает новые учебные системы программирования «БЕТТАЛОГО» и «АЛТ-ПАСКАЛЬ» для УКНЦ, «ЛОГО» и «РОБОТЛАНДИЯ» для классов БК-11, «ТУРБО-ПАСКАЛЬ» для КУВТ-86. Делаются крупные вложения в создание автоматической системы составления расписаний для школ, ПТУ и техникумов. НТП «Новая Альтернатива» резко расширяет объем разработок и продаж.

Адрес: 656057, г. Барнаул, а/я 2513,
НТП «Новая Альтернатива»,
директор Гриценко Алексей Николаевич.

Региональная программа

В целях обеспечения должного уровня подготовки кадров, способных принимать активное участие в формировании информационного общества, в рамках процесса информатизации экономики и различных отраслей народного хозяйства (общества в целом), Управление образования администрации Самарской области предлагает в соответствии с базисным учебным планом Министерства образования Российской Федерации в рамках регионального компонента (учебных часов, отводимых на усмотрение региона) реализовать учебный курс «Информационная культура», с одной стороны, ликвидирующий дефицит учебного времени, отводимого на изучение стандартного курса ОИВТ, с другой стороны, создающий условия для серьезной подготовки учащихся в области использования компьютерных технологий и компьютерных средств.

Е. Я. Коган,
г. Самара
Ю. А. Первин,
г. Переславль-Залесский

КУРС «ИНФОРМАЦИОННАЯ КУЛЬТУРА» — РЕГИОНАЛЬНЫЙ КОМПОНЕНТ ШКОЛЬНОГО ОБРАЗОВАНИЯ

Авторы курса «Информационная культура» — предприятие «Роботландия», ассоциация «Компьютер и детство», Институт проблем информатики Российской академии наук, Институт новых информационных технологий в образовании, предприятие «ИнфоМир» и ряд других научных коллективов и специалистов Москвы, Самары и Переславля-Залесского. Этот курс внедряется с 1994/95 учебного года в школах Самарской области.

В современных условиях к молодому поколению, вступающему в жизнь, предъявляются иные требования, чем четверть века назад. Молодой человек должен обладать знаниями в тех областях науки и техники, которые за последнее десятилетие оформились в виде научных дисциплин. В еще большей мере ему нужны умения и навыки планирования своей деятельности, поиска информации, нужной для решения стоящих перед ним задач, проектирования и построения информационных моделей, дисциплины общения и структурирования сообщений, инструментирования всех видов деятельности, так же как и навыки использования современных технических средств в жизни, быту, творчестве, на производстве.

Совокупность всех этих новых знаний, умений и навыков составляет **операционный стиль мышления**, формирование которого — задача современной школы, вооруженной для этого концептуальным фон-

дом информатики и современной вычислительной техникой.

Умея работать с необходимыми в повседневной жизни вычислительными и информационными системами, базами данных и электронными таблицами, персональными компьютерами и информационными сетями, человек информационного общества приобретает не только новые инструменты деятельности, но и (это главное!) новое видение мира. Культурный уровень такого современного молодого человека характеризует понятие **информационной культуры**, которая в силу фундаментальности ее составляющих должна формироваться в школе начиная с первых уроков.

Проблема формирования информационной культуры у целого поколения людей, вступающих в жизнь, представляет социальный заказ современного информационного общества. В настоящее время складываются определенные предпосылки для ее решения, а именно:

- персональный компьютер перестает быть экзотикой в школе и превращается в орудие повседневной учебной деятельности;
- накоплен опыт обучения детей разных возрастов основам информатики и использованию компьютера на предметных уроках;
- система образования получила известную свободу в построении учебных программ и учебных планов;
- в школе появилось понятие *регионального школьного компонента* (регионального часа), позволяющее ориентировать учебно-воспитательный процесс на наиболее актуальные (*для региона)* образовательные задачи.

Таким образом, в настоящее время имеется реальная возможность реализовать сквозной курс «Информационная культура» (от I до выпускного класса) на базе предоставляемого школам регионального часа.

В данной статье предлагается подход к построению и реализации такого курса, который с начала 1994/95 учебного года внедряется в общеобразовательных школах Самарской области.

Что же представляет курс «Информационная культура», каковы его методологические, педагогические, организационно-технические возможности?

Проектируя курс «Информационная культура», нельзя не считаться с реальными условиями, в которых он будет внедряться в школу. Ниже рассмотрены некоторые требования к курсу. Хотя большинство их определено конъюнктурными условиями сегодняшнего дня, нельзя не признать объективность этих требований. Обсуждаемые требования объединены в несколько групп (классификация требований достаточно условна).

Методологические требования

1. Речь идет об обязательном курсе в общеобразовательной школе. Поэтому в нем исходно отвергается профессиональная ориентация в пользу мировоззренческого направления. С этой точки зрения **основной задачей курса признается формирование стиля мышления, адекватного требованиям современного информационного общества.**

2. В дискуссии о процедурном и функциональном подходе к изучению информатики (стержневой дисциплине курса) «Информационная культура» должна придерживаться золотой середины: знакомство с реальными пакетами прикладных программ, конечно, имеет место, но не ценою снижения уровня изучения фундаментальных понятий и структур информатики.

3. Знания и умения, полученные в курсе, станут арсеналом искусства проектирования в широком смысле этого слова (в отличие от узкопонимаемого ремесленнического программирования). Технологии проектирования (в частности, конструктивная процедурная технология «сверху вниз») — основная содержательная оболочка многих тем курса «Информационная культура» в разные годы обучения.

4. Курс «Информационная культура» — обязательная дисциплина массовой общеобразовательной школы. Никакие элитарные ограничения при внедрении этого курса в школу не допускаются. Курс спроектирован не для элитарной, а для

обычной общеобразовательной школы. Он доступен для усвоения школьнику средних способностей.

Поиск талантливых учеников и работа с ними должны выполняться на уровне индивидуального подхода учителя к детям.

5. В курсе максимально возможно реализуются коллективные формы обучения (информатизация образования открывает здесь широкие возможности). Одна из задач курса — воспитать ответственность за принимаемое решение перед коллективом. С другой стороны, организация обучения, и в частности компьютерных упражнений и проектов, демонстрирует ученику возможности черпать значительные ресурсы из коллективного решения задачи.

6. Компьютер предстает перед детьми как универсальная информационная (а вовсе не вычислительная) машина. Курс высвечивает роль компьютера как инструмента, эффективно совершенствующего и организующего общение людей.

7. Прямо и косвенно курс обязан работать на гуманитаризацию образования. Рассматривая разнообразные информационные процессы, сквозной курс «Информационная культура» изучает структуры литературных и музыкальных произведений, показывает в компьютерном рисунке не только типовые элементы графической информации, но и выразительные средства, и содержательное наполнение.

8. Программный инструментарий курса вместе с многообразием форм учебного процесса призван обеспечить исследовательскую и творческую его направленность. Дети учатся первым навыкам проведения, оформления и защиты научного исследования. Средствами редакторов информации, баз данных, издательских систем ученики печатают стенгазеты, выпускают сборники сочинений, организуют вернисажи.

9. Принцип дидактической спирали является одним из факторов структуризации в методике большинства предметных дисциплин: сначала изучение фундаментальных понятий на этапе начального образования, затем их последующее развитие в центральных классах средней школы, наконец, научное обобщение в старших (выпускных) классах. Та же спираль наблюдается и в курсе «Информационная культура»: сначала простейшие исполнители непосредственного управления, затем компьютер, выступающий в роли устройства, управляющего исполнителем, наконец, программные средства управления.

Педагогические требования

10. Об идеале информатизации школьного образования можно было бы говорить при кардинальной перестройке содержания обучения и методики преподавания всех школьных предметов. Такая постановка задачи потребовала бы сегодня больших затрат времени и ресурсов.

Поскольку сквозной курс «Информационная культура» проектируется в условиях дефицита времени, принято упрощающее соглашение: не считать требование учета межпредметных связей первоочередным. Это соглашение существенно раскрепощает проектантов и позволяет реализовать проект в разумные сроки. Следует, однако, сказать, что фактор межпредметных связей тем не менее оказал свое влияние на предлагаемый курс по двум причинам: во-первых, педагогический стереотип мышления разработчиков не позволил им в полной мере воспользоваться предоставленной свободой, а во-вторых, зачастую межпредметные связи настолько органичны, что отказ от их демонстрации означал бы, наоборот, усложнение, а не упрощение разрабатываемой темы, урока, фрагмента.

11. Среди школьных предметов, примыкающих к курсу наиболее близко, следует безусловно назвать информатику.

Поэтому одним из важнейших вопросов внедрения курса «Информационная культура» является вопрос о его соотношении с типовым школьным курсом информатики. Решение этого вопроса динамично. В ближайшие два—четыре года (переходный период), по нашему мнению, оба курса будут сосуществовать.

С завершением внедрения курса во всех классах одиннадцатилетнего среднего образования необходимость в отдельном

курсе информатики может быть снята. Опасения, связанные с тем, что во многих высших учебных заведениях в настоящее время появился внутренний вступительный экзамен по информатике, не имеют оснований, поскольку объем знаний, которые школьники получают при систематическом обучении в курсе «Информационная культура», покрывает требования, предъявляемые к абитуриентам практически во всех высших учебных заведениях.

12. Учитывая самостоятельную значимость курса «Информационная культура», его оригинальное содержание и нетрадиционную сквозную организацию, следует считать обязательным требование методической поддержки курса оригинальными учебниками, пособиями, дидактическими материалами. Рекомендацию воспользоваться существующими школьными пособиями или пользовательскими инструкциями существующих программно-методических систем нельзя считать пригодной даже в качестве временного решения.

13. Важнейший фактор, учитываемый в проекте курса — это преподавательский контингент, которому предстоит внедрять этот курс в школу.

Этапы начального образования (I—III, IV классы) должен вести тот же учитель (классный руководитель), который преподает остальные основные предметы.

Следовательно, в методических пособиях для учителя учитывается специфический уровень подготовки учителя начальной школы, в который не входят знание программирования, операционной системы и профессиональных прикладных программ. Пособия для учителей начальной школы — подробные и доступные.

14. Уроки по курсу информационной культуры в средней школе следует поручать учителю информатики. С одной стороны, этим обеспечивается требуемый уровень квалификации. С другой стороны, его участие в таком курсе будет способствовать повышению авторитета учителя информатики и его предмета. Наконец, повышение нагрузки за счет предмета, в котором у учителя есть полная возможность проявить свою творческую индивидуальность, послужит моральным и материальным стимулом.

15. Все компоненты сквозного курса информационной культуры ориентированы на традиционную классно-урочную систему. Вместе с тем, поскольку уроки этого курса проводятся в основном в школьном кабинете информатики с 10—15 компьютерами, необходимо воспользоваться той же административной документацией, которая разрешает делить класс на подгруппы при проведении уроков информатики в компьютерном кабинете.

Организационные требования

16. Курс состоит из 11 частей-модулей, каждая из которых соответствует одному учебному году. В курсе предусмотрено несколько так называемых **точек входа** — тех модулей (учебных лет), с которых может начинаться курс.

Разумеется, если точка входа отнесена далеко от I класса, то учителю, начинающему курс, придется уделить определенное внимание восполнению пробелов. Учебники и пособия для классов, являющихся точками входа, предусматривают разные варианты вхождения учащихся в курс. При этом признается целесообразным написание двух (или трех) комплектов методических материалов в зависимости от точки входа. В одном учебнике должны быть учтены обсуждаемая вариативность за счет двух вариантов нескольких первых параграфов, в которых излагаются тезисы предшествующих модулей, необходимые для восприятия курса от заданной точки входа.

17. Объем каждого годового модуля определяется понятием **регионального часа**. Один региональный час в неделю формирует годовой курс в 32 часа. Таким образом, общий объем десяти-одиннадцатилетнего курса «Информационная культура» равен 320 (352) часам.

18. Несомненно, самый сложный, ответственный и трудоемкий этап внедрения курса — переподготовка учителей. Одним из первых методических документов, создаваемых по проекту «Информационная

культура», является подробно разработанный **учебно-методический план** курсов повышения квалификации учителей, предусматривающий осуществление этой деятельности в рамках регионального института повышения квалификации с участием авторов методических систем и программных разработок.

19. Основной рабочий кабинет — кабинет информатики. Практика показывает, что необходим еще один дополнительный кабинет. Два кабинета нужны еще и потому, что школьники младших классов и старшеклассники должны работать в различных эргономических условиях. Становление курса информационной культуры в начальной школе делает проблему кабинета информатики для младших школьников особенно острой.

20. Для учащихся, которые изучают информационную культуру в школе, полезно предусмотреть их летнее участие в детских лагерях, где они смогут наряду с отдыхом еще раз встретиться с компьютерами, информационными и программными системами в новых, как правило, более комфортных условиях.

Такие летние лагеря (или специализированные смены) полезно предусмотреть особенно в первые годы становления курса.

Программа внешкольной работы с детьми в области информационной культуры должна быть разработана отдельно и сопрягаться с содержанием курса «Информационная культура».

Технические требования

21. В программе курса сочетаются адаптированные учебные и профессиональные программные средства. В младших классах преобладают оригинальные учебные программы, а в обучение выпускников наряду с адаптированными средствами включается знакомство с рядом профессиональных информационных и программных систем.

Впрочем, использование профессиональных программных средств школьниками не означает ориентацию их на изучение программной документации. Методические пособия и в этом случае подготавливаются специально для учащихся массовой общеобразовательной школы.

22. Тип компьютеров, используемых в курсе, задан изначально. Это компьютеры PC и совместимые с ними. Минимально допустимая конфигурация: оперативная память не менее 640 килобайт, мониторы с адаптерами не ниже EGA (для ряда отечественных PC-совместимых машин, в частности компьютеров с мониторами CGA эти условия не удовлетворяются).

23. Роль компьютеров в организации коммуникационных функций информационного общества фундаментальна, и проблемы компьютерных коммуникаций не могли не найти свое отражение в курсе.

В классе весьма желательна локальная сеть. Важно установить в школах региона модемы для обеспечения компьютерных коммуникаций по глобальным сетям (или во всяком случае по региональной сети системы образования). Поэтому при изучении задач коммуникации в курсе используются средства моделирования локальных сетей.

24. Принцип моделирования реальных ситуаций реализуется в ряде тем в старших классах: допуская отсутствие модемов в школах и возможностей выхода в глобальные информационные сети, курс должен тем не менее обосновать актуальность информационных коммуникаций и сформировать начальные навыки межкомпьютерного общения.

Общая структура сквозного курса «Информационная культура» (в рамках регионального часа)

I класс

Компьютер — твой друг. Элементарное введение в практику общения с компьютером. Компьютер на уроках математики и русского языка.

Простейшие тренажеры «клавиатурного» и «мышиного» интерфейса. Упражнения на прямой и обратный счет, состав числа и простейшую арифметику. Компьютерные упражнения с программами систем «Малыш», «Путешествие в страну Букварию» и «Роботландия».

Для модуля подготовлены книжка-раскрашка «Компьютер — твой друг», методическое пособие для учителя и сборник пользовательских инструкций к программам.

II класс

Множества, признаки и порядок. Первые представления о множествах, подмножествах и элементах. Закономерности и порядок. Классификация и конструирование. Компьютер на уроках математики и русского языка.

Игры и упражнения на классификацию, тренировку памяти и конструирование, базирующиеся на программах пакетов «Классификаторы» и «Малыш».

Для модуля подготовлен иллюстрированный задачник «Вычислялки, соображалки, запоминалки» с упражнениями на классификацию, анализ характеристических признаков множества, конструирование и тренировку памяти. В составе методического обеспечения — учительское пособие в виде поурочных методических комментариев.

III класс

Алгоритмы. Алгоритмы вокруг нас. Компьютерные модели.

Рассматриваются и подробно обсуждаются понятия команды и алгоритма. Упражнения с программами из разделов «Алгоритмические этюды» системы «Роботландия+», пакетов «Малыш» и «Арифметика». Инструментарий компьютерных уроков математики.

Для модуля готовится книга для чтения на основе материалов «Роботландии», а также учительское пособие.

IV класс

Компьютер помогает думать. Понятие об исполнителях и способах управления. Распознавание закономерностей и исследование алгоритмов, метод «черного ящика».

Общая схема изучения исполнителей. Пропедевтика управляющих структур. Решение комбинаторных задач и конструирование исполнителей. Упражнения с программами из раздела «Исполнители» системы «Роботландия+» и пакета «Веселые картинки».

Для модуля готовится книга для чтения и учительское пособие.

V класс

Компьютер и слово. Текстовый редактор: диктанты, сочинения, стенгазеты и книги.

Рассматриваются механизмы обработки текстовой информации и возможные в школе приложения адаптированного текстового редактора как в рамках курса информационной культуры, так и на других предметах.

Для модуля подготовлена книга для чтения «Компьютер и слово» на основе пакета программ «Торт», а также методическое пособие и сборник пользовательских инструкций для учителя.

VI класс

Кукарача и Черепашка учатся у компьютера. Процедуры, параметры, рекурсии.

Исполнитель Кукарача. Непосредственное и программное управление. Процедуры. Редактирование процедур. Управляющие структуры языка управления Кукарачей.

Лого — язык детей и компьютеров. Начальное графическое введение в язык программирования Лого. Первые представления о технологии проектирования. Структуры данных, рекурсии и мультипликации.

По модулю готовится учебник для школьников — элементы программного управления Кукарачей и Черепашкой Лого, первые уроки программирования. Отдельная книга — методическое пособие для учителя.

Программные средства этого модуля — программно управляемый исполнитель Кукарача из «Роботландии+» и русифицированный вариант системы Logo-Writer.

VII класс

Компьютер — инструмент искусства. Графические редакторы, компьютерные вернисажи. Редактирование музыкальной информации.

Изучается пакет адаптированных графических редакторов с использованием их на уроках и во внеклассной работе. Применение графики в прикладных системах (конструирование сказок).

Вводятся начальные сведения из музыкальной грамоты и редактирования музыкальной информации. Проведение выставок, турниров, концертов в школьном кабинете информатики.

Для модуля подготовлена книга для школьника по графическому и музыкальному редактированию и учительское методическое пособие.

VIII класс

Информационные хранилища. Банки данных и электронные таблицы.

Обсуждаются проблемы организации хранения и поиска информации. Понятие запроса и языка управления базами данных. Применение баз данных в учебной и внеклассной работе.

Что такое банк и безналичные деньги, что должно быть в информационной системе о домашней (школьной) библиотеке.

Приемы работы с электронными таблицами.

Для модуля готовится книга для чтения по базам данных и электронным таблицам, а также методическое пособие для учителя. Должно

быть разработано оригинальное программное обеспечение в составе двух исполнителей, адаптированной базы данных и адаптированной электронной таблицы.

IX класс

Кодирование информации.

Начало модуля посвящено восполнению пропущенных навыков — повторению основных понятий алгоритмизации. Изучаются способы кодирования без компьютеров (системы счисления, координаты и т. д.) и на компьютере (двоичное кодирование, способы упаковки и шифрования информации). Обсуждается устройство памяти компьютера и структура хранения информации.

По курсу подготовлен учебник для школьника в виде хрестоматии по информатике для двух лет обучения (IX, X классы) и учительское пособие.

X класс

Информационные модели.

Основной инструмент модуля — гипертекст. Дается представление о способах передачи информации между людьми и между компьютерами, разбирается технология пересылки информации по модему. Основная часть модуля посвящена построению информационных моделей — от простейших (модель кинозала, кодирование геометрической информации) до производственных (база данных и графический редактор). Рассматривается работа с системой геометрических построений ПланиМир и ее внутреннее устройство.

Программное обеспечение — система КуМир-93 и комплект гипертекстовых практикумов по темам уроков — разрабатывается предприятием ИнфоМир.

XI класс

Современные информационные технологии. Информационный мир. Издательские системы. Телекоммуникации. Экспертные системы. Мультимедиа-технологии. Электронные энциклопедии.

Дается представление о современных информационных технологиях. Демонстрируются принципы построения гипертекстовых энциклопедий, а также приемы работы с ними. Упражнения по подготовке материалов в настольных издательских системах. Понятие о коммуникациях в локальных и глобальных информационных сетях. Работа с экспертными системами. Обработка видео- и аудиоинформации средствами мультимедиа.

Организационная структура курса

Курс «Информационная культура» представляет собою систему модулей, каждый из которых соответствует одному году обучения. С точки зрения дидактического содержания обучения в последовательности модулей выражена та дидактическая спираль, которая характерна для других естественнонаучных циклов в системе школьного образования: фундаментальные понятия в элементарном изложении на начальном этапе образования; возвращение к ним на более высоком методическом уровне в старших классах. Это обстоятельство позволяет внести в систему модулей определенную автономность.

Во-первых, учащиеся, покидающие общеобразовательную школу с незавершенным средним образованием, получают тем не менее основы информационной культуры в том объеме, который позволит им ориентироваться в окружающей их обстановке информационного общества за рамками школы.

Во-вторых, появляется возможность организовать учебный процесс по курсу не в строгой последовательности, начинающейся в I классе и завершающейся в XI, а предусмотреть несколько возможных отправных пунктов для начала его изучения — так называемых точек входа.

В-третьих, модульная структура курса позволяет организовать этапную разработку проекта с относительно четким разделением функций между отдельными авторскими коллективами разработчиков.

Возможные точки входа в курс и модификации учебно-тематического плана

Базовый вход.
Первая точка входа — I класс
Временная точка входа — II класс
Вторая точка входа — V класс
Третья точка входа — IX класс

Базовый вход. Первая точка входа

В современном школьном образовании существуют две модификации учебного плана для начальной школы: 1—3 и 1—4.

Для модулей III—IV классов предлагается выпустить учебник под одной обложкой. В зависимости от принятой в конкретной школе схемы обучения (10 или 11 лет)

учителю рекомендуется принять полный или уплотненный учебный план. Приводимые в качестве приложения к учительскому пособию эти два учебно-тематических плана III—IV классов помогут учителю в настройке на необходимую схему обучения.

При этом модульная структура курса позволяет не подвергать корректировке весь последующий материал базового плана.

Временная (вспомогательная) точка входа во II классе призвана упростить организационные проблемы, связанные, с одной стороны, с широким внедрением курса в школах большого региона, а с другой стороны, со значительным временным разрывом между началом обучения в I классе и переходом в среднюю школу (V класс). Школьникам-второклассникам, приступающим к изучению курса, предстоит преодолеть относительно небольшое восполнение пропущенных навыков. Задача тем более упрощается, что в той большой работе по обучению грамоте, которая проводится с помощью компьютеров у первоклассников, здесь, по существу, нет необходимости: контингент II класса — это, как правило, читающие ученики. Многие программы, требующие восприятия текстовой информации, могут работать в модуле II класса без дополнительной адаптации.

Вторая точка входа

Школьники начинают знакомство с информационной культурой непосредственно с обработки текстов — основного в современном мире вида компьютерной обработки информации. Необходимые понятия алгоритма, исполнителя, команды, систем команд могут быть восполнены несколько позднее (в IX—X классах) на очередном витке дидактической спирали.

В организационном плане V класс представляет собою очень удобную точку входа. В связи с переходом в среднюю школу с ее многопредметным образованием появляется возможность поручать ведение курса наиболее подготовленной категории преподавателей — профессиональным учителям информатики.

Третья точка входа

Предварительные требования к учащимся, которые приходят в IX класс, с точки зрения их подготовленности к восприятию курса «Информационная структура»: минимальный опыт работы на ЭВМ, знакомство с клавиатурой, опыт составления программ длиной 10—20 строк на любом языке.

Эти требования могут быть сняты ценой проведения 8 дополнительных уроков и уменьшением объема учебных занятий, предложенного в соответствующем учебно-методическом плане на 8 часов.

В конкретных условиях реализации курса пришлось пойти на некоторые отклонения от описанной здесь схемы. В частности, третья точка входа перенесена с сентября 1994 г. на следующий учебный год.

А. Р. Белкин,
старший научный сотрудник Института автоматизации и проектирования РАН
С. И. Шумов,
старший научный сотрудник Института автоматизации и проектирования РАН

АНАЛИЗ И ОЦЕНКА ТРАДИЦИОННЫХ И НЕТРАДИЦИОННЫХ МЕХАНИЗМОВ ПОЛУЧЕНИЯ И ОБОБЩЕНИЯ НОВЫХ ЗНАНИЙ*

Разделение и сопряжение традиционных и нетрадиционных знаний

I. Переработка информации и получение знаний

Традиционные научные представления о накоплении знаний выглядят в виде следующей достаточно хорошо известной многоэтапной схемы.

1. Сбор и накопление информации об окружающем мире.

2. Выдвижение гипотезы (или системы гипотез), объясняющей ранее непонятные факты.

3. Исследование выдвинутой гипотезы, анализ ее следствий, поиск противоречий.

4. Целенаправленный поиск информации (фактов), подтверждающей или опровергающей выдвинутую гипотезу. Экспериментирование.

5. Превращение экспериментально подтвержденной гипотезы в теорию (концепцию, парадигму). Использование ее для расширения и дополнения теоретической картины мира и для построения новых теорий.

На первом этапе получение информации осуществляется фактически эмпирическим путем на основании использования обычных органов чувств (применение научных приборов не вносит принципиальной разницы, ибо фактически лишь раздвигает границы и усиливает возможности органов чувств).

Второй этап предполагает способность человека к индуктивному обобщению собранных фактов, нередко реализующуюся в виде так называемого инсайта — озарения, при котором собранная информация

---

* Окончание. Начало см.: Информатика и образование. 1994. № 6.

предстает под новым, неожиданным углом. Природа собственно инсайта практически мало изучена, хотя есть основания полагать, что основную роль при этом играет деятельность подсознания.

На последующих этапах человек применяет свои дедуктивные, логические способности. Разумеется, на каждом этапе возможен возврат к одному из предшествующих, и весь процесс далеко не во всех случаях доходит до своего логического завершения.

Таким образом, процесс получения и пополнения научного знания укладывается в следующую схему:

Эмпирика — Индукция — Дедукция

Заметим, что помимо традиционно научной существует целый ряд иных схем (хотя получаемый в результате их «продукт» традиционная наука знаниями не считает).

Для обыденного же, рассудочного, мышления четвертый и пятый этапы (и частично третий) обычно отсутствуют, а поиск логического обоснования заменяется фиксацией более или менее случайных совпадений или взаимосвязей (пространственно-временных, дополнительных или, реже, каузальных), соединяющих разнородные факты. Индукция при этом в значительной степени заменяется или дополняется традукцией (рассуждением по аналогии), так что весь процесс может быть представлен в виде:

Эмпирика — Индукция, Традукция

«Знания», получаемые в итоге такого процесса, обычно не составляют теории, но предстают в виде правил (типа «сунешь палец в огонь — обожжешься»), примет («ласточки летают низко — к дождю»), житейской мудрости и т. п. Область их применения ограничена, применимость часто небесспорна, выводы и рекомендации нередко не выдерживают экспериментальной проверки. В то же время такие «обыденные знания» играют важную роль в повседневной жизни человека и часто могут выступать в роли своего рода «питательной среды», «фона», обработка которого научными методами приводит к получению подлинно научных знаний.

Для мышления религиозного познание мира эмпирическим путем представляется сравнительно маловажным и на первый план выступает получение так называемых откровений, суть которых — эзотерическое постижение окружающего мира не посредством органов чувств, но непосредственно «душой». Откровения могут содержать в себе как отдельные «факты», касающиеся природы мироздания, так и целые более или менее стройные теории, объявляемые при этом боговдохновенными. Естественно, итоги эзотерических откровений не подлежат ни ревизии, ни какой бы то ни было экспериментальной проверке. Более того, в догматических религиях (типа раннего христианства) недопустим даже чисто логический, дедуктивный поиск следствий из боговдохновенных теорий; если же такой поиск проведен и привел к противоречиям, последние попросту игнорируются*. В иных религиях (например, в иудаизме), подобный поиск следствий вполне допустим, причем применяемый аппарат может быть весьма изощрен, включать в себя идеи логики предикатов и модальной логики, диалектические методы, дедуктивные и традуктивные рассуждения и пр. Примерная схема получения религиозных «знаний» может, таким образом быть представлена в следующем виде:

Эзотерика — Индукция, Традукция, Дедукция

Наконец, оккультно-мистическое мышление допускает как эмпирический, так и эзотерический путь получения ис-

* Тертуллиану, как известно, принадлежит соответствующий лозунг: Credo quia absurdum! (Верю, ибо нелепо! — *лат.*) — *Примеч. автора.*

ходных фактов, индуктивное и традуктивное выдвижение гипотез, их экспериментальную проверку (хотя довольно часто недостаточно тщательную и всестороннюю) и дедуктивный анализ, что укладывается в схему типа:

Эзотерика,
Эмпирика — Индукция,
Традукция — Дедукция

С мышлением религиозным его, однако, роднит то, что обнаруживаемые в теории противоречия часто замалчиваются или искусственным путем обходятся. Кроме того, здесь также допускается получение эзотерическим путем не только исходной информации, но и вторичных теорий и целых парадигм, что беспристрастному исследователю может показаться чрезмерным. Тем не менее, отказавшись от этого и сведя до минимума роль традуктивных рассуждений, можно прийти к некоей рафинированной научно-мистической схеме типа:

Эзотерика, Эмпирика —
Индукция — Дедукция,

представляющей несомненный интерес. Фактически речь может идти в первую очередь о том, чтобы принять возможность не только эмпирического, но и эзотерического пути получения некоторой исходной информации об окружающем мире.

II. Обоснование эзотерического пути получения информации

Исследуя эзотерический путь получения информации, следует отдельно обратить внимание на вопрос об источнике информации, поступающей к нам подобным образом, и о способах восприятия ее человеком.

Для субъекта, склонного к религиозному или религиозно-мистическому мироощущению, ответ на первый вопрос очевиден и восходит к вмешательству неких высших сил, обычно наделяемых определенной индивидуальностью и так или иначе персонифицируемых. Так же очевидно, что объективный исследователь, желающий сохранить беспристрастность, вынужден отклонить гипотезу о подобном воздействии как разновидность deus ex machina, считая ее недостаточно подтвержденной.

Следуя основополагающему принципу Оккама и не пытаясь «умножать число сущностей сверх необходимого», имеет смысл интерпретировать факты получения человеком эзотерической информации на той же методологической базе, как и факты получения информации эмпирической, а именно как *результат воздействия на наши органы чувств некоторой объективной реальности и последующего осознания и осмысления этого воздействия*.

Для более подробного анализа здесь имеет смысл упомянуть, что, с оккультно-мистической точки зрения, человек живет не только в окружающем его физическом мире, но также и в двух других, «высших» мирах: астральном и ментальном*. В обычном физическом материальном мире действует тело человека, душа его действует в мире астральном, ментальный же мир — прибежище человеческого разума. Сверх того, оба «высших» мира обычно населяются множеством разнообразных «астральных» и «ментальных» существ, но это предположение серьезный скептический исследователь вполне может игнорировать, памятуя о склонности человека персонифицировать непонятные и/или пугающие его явления природы**.

Ограничимся менее обязывающими допущениями о том, что действительно человек одновременно существует в трех взаимосвязанных планах. Физический мир описывается известными физическими (точнее, естественнонаучными) представлениями и включает в себя неживую при-

---

* В каком смысле эти миры «выше» физического, обычно не уточняется. — *Примеч. автора.*

**Лавуазье приписывается известное высказывание на эту тему: «Если бы древние греки осознавали силу тяжести как природное явление, среди богов-олимпийцев восседал бы и бог тяжести!» — *Примеч. автора.*

роду, флору, фауну и человека. Мир астральный включает в себя так называемый астральный прообраз человека, в роли которого выступает подсознательная сфера (разница между душой и подсознанием в данном случае выглядит лишь терминологической). Вполне допустимым выглядит предположение о том, что астральные прообразы могут существовать также у высших животных (в частности, приматов) с развитой психикой.

Принимая в качестве постулата предположение о принципиальной познаваемости астрального мира, логично допустить, что астральный мир может быть описан при помощи каких-то своих законов, отличных от физических, но тем не менее вполне постижимых. В частности, не следует исключать предположений о том, что в нем могут существовать некоторые природные явления, воздействующие на указанные объекты астрального мира, а также что сами эти объекты также могут воздействовать друг на друга. Механизм этого воздействия, а равно отражение его в физическом мире весьма мало изучены*.

Все сказанное в определенной мере может быть отнесено и к ментальному миру, включающему ментальный прообраз человека (а равно и иных мыслящих тварей, буде таковые обнаружатся).

С учетом сказанного выше эзотерическое получение информации человеком может быть **логично** объяснено, как результат воздействия, произведенного на его астральный либо ментальный прообраз, эмпирическая же информация поступает через обычные, «физические» органы чувств. В первом случае информация поступает в виде некоторого смутного ощущения, во втором же возможно поступление информации на понятийном и даже вербальном уровне.

В связи с тем что ментальные феномены чрезвычайно редки (и, строго говоря, даже факт их существования весьма спорен), здесь мы ограничимся лишь рассмотрением **воздействий** на астральный прообраз, из которых упомянем наиболее известные.

Ощущение чужого взгляда

Широко известный и неоднократно описанный в литературе феномен. В направлении взгляда, по мнению оккультистов, излучается мощный поток так называемой астральной энергии (смысл этого термина обычно никак не объясняется), ощущаемый на значительном расстоянии. Так или иначе, ряд источников указывает, что взгляд тренированного индуктора в лицо ощущается тренированным реципиентом на расстоянии до 25—30 м, взгляд в спину — до 7 м.

Широко известен аналогичный феномен с животными, не выносящими пристального взгляда прямо в глаза.

Ощущение чужого присутствия

Также хорошо описанный, хотя и мало изученный феномен, объясняемый оккультистами восприятием чужой «астральной энергии»... Животные, особенно домашние (собаки, кошки), также считаются в быту весьма чувствительными к появлению в доме чужого человека или иного незнакомого астрального объекта.

Непосредственное восприятие чужих эмоций

Непосредственное восприятие чужих эмоций (эмпатия), в отличие от ментального феномена телепатии (восприятия чужих мыслей), — феномен, знакомый многим и хорошо известный. Оккультисты объясняют его тем, что возбужденный, взволнованный человек излучает «астральную энергию» гораздо интенсивнее и она легче воспринимается окружающими людьми.

Животные также хорошо воспринимают сильные эмоции хозяина и ведут себя соответственно. Отметим, что возможен и обратный эффект: необычная возбужденность и странности поведения животных, например в преддверии землетрясения, легко замечаются и отмечаются обычными людьми, далекими от этологии.

Итак, эзотерическая информация, поступающая к человеку, может быть следствием оказанного на его астральный прообраз воздействия со стороны иных объектов

* Часто используемый термин «информационное поле» является, в сущности, только лишь ничем серьезным не подтвержденным термином. — *Примеч. автора.*

астрального мира. Логично предположить при этом, что для восприятия поступающей информации человек располагает некоторым специфическим органом (или даже комплексом органов) чувств, воспринимающим астральные воздействия, который обычно, следуя Смайгренду*, называют внутренним или астральным слухом.

Согласно концепции Смайгренда, основное отличие астрального слуха от обычных органов чувств состоит в том, что его сигналы осознаются человеком не непосредственно, но опосредованно, через подсознание и поэтому чаще всего просто «забиваются» огромным потоком информации, непрерывно поступающей в мозг от обычных органов чувств, прежде всего зрения и слуха. Лишь в условиях *сенсорной депривации*, когда поступающей в мозг информации становится недостаточно, чувствительность органов чувств обостряется и подсознание «выдает» наверх дополнительные данные, поступившие через внутренний слух. Косвенным подтверждением сказанному служит тот факт, что перечисленные феномены проявляются в основном в условиях недостаточности обычной информации (так, например, эффект чужого присутствия неподалеку чаще всего ощущается в темной тихой комнате, т. е. в условиях зрительно-слуховой депривации).

Отметим здесь, что у животных, лишенных сознания, обычные органы чувств и внутренний слух оказываются «уравненными в правах», в связи с чем внутренний слух оказывается сравнительно куда более чутким, чем у человека. В целом концепция Смайгренда выглядит вполне логичной и не содержит внутренних противоречий.

III. Эволюция представлений о природе сознания и реальности

Долгие годы научные и мистические представления о реальности противопоставлялись друг другу, считались взаимоисключающими. Естественно, при этом право на истину признавалось только за научным описанием мира.

Сейчас наши представления о реальности становятся менее жесткими. Мы начинаем понимать, что мир бесконечно разнообразнее и сложнее любых наших представлений о нем и право на существование имеют самые различные формы описания реальности, которые не исключают, а дополняют друг друга, описывая просто разные аспекты реального мира. С этой точки зрения и научные, и мистические представления являются такими дополняющими друг друга формами описания реальности. И лишь теперь развитие современной науки, ее концептуального аппарата позволяет говорить о точках соприкосновения между ними.

Взаимосвязь научного и мистического описания мира можно рассмотреть на примере одной очень важной проблемы — представлений о природе окружающего нас пространства, его взаимосвязи с разными уровнями организации человека.

Окружающая нас реальность, как полагают оккультисты, имеет несколько планов. Помимо физического мира плотных форм, воспринимаемых зрением, слухом и другими органами чувств, рассматривается мир тонкоматериальный, который не доступен обычным органам чувств, но не менее реален, чем первый. Собственно, все древние формы шаманизма, колдовства, магии ориентировались на установление контакта с этим тонкоматериальным, астральным миром и использование его сущностей и энергий для решения самых разных практических задач. Соответственно и человек рассматривался как одновременно существующий в нескольких планах реальности.

До недавнего времени наука категорически отрицала обоснованность — какую бы то ни было — подобного взгляда на мир. Однако развитие современной физики, психологии, нейрофизиологии, парапсихологии показывает, что реальный мир может быть устроен гораздо ближе к этим представлениям, чем к примитивной механистической схеме, описывающей лишь мир плотных форм.

Собственно говоря, в представлении о

* Здесь мы следуем в основном канве наиболее полной работы: *Smygroend J.* Sub-Conscience and the Inner Ear. NY, 1985, хотя она далеко не единственна у данного автора. — *Примеч. автора.*

том, что вокруг нас существует огромный пласт реальности, недоступный непосредственному восприятию обычных органов чувств, нет ничего мистического. Вспомним, что наше тело буквально пронизано различными волнами электромагнитного диапазона, через него постоянно пролетают элементарные частицы, приходящие из космоса, на него воздействуют радиационные поля земного происхождения, акустические волны ультразвукового и инфранизкого диапазонов — и все это несет громадные потоки информации, незримо воздействует на наш организм (достаточно вспомнить хотя бы влияние солнечной активности). В реальности этого мира убедиться несложно — достаточно включить телевизор или радиоприемник. Но если бы кто-нибудь стал говорить о реальности подобных незримых и неслышимых сигналов до изобретения радио — его обвинили бы в мистицизме и прочих грехах.

Чтобы признать возможным существование тонкоматериального мира, ученый может допустить возможность образования устойчивых, длительно сохраняющихся структур, например, волнового, вихревого типа. Такие вихри представляют собой энергетические структуры, не связанные жестко с веществом (атомами, молекулами), как окружающие нас обычные материальные объекты. Условия возникновения и развития подобных вихрей и других более сложных образований изучает новая область науки — **синергетика**, направленная на исследование процессов самоорганизации в живой и неживой природе, выявление наиболее общих закономерностей самоорганизующихся систем.

Заметим, что индийские тексты по йоге содержат в себе описания тонкоматериальных центров человеческого тела — чакр, имеющих форму конусообразных вихрей, с вершинами на позвоночнике, а основаниями круглой формы на передней части тела. Сенситивы, описывая чакры, говорят об ощущении вращения, а само слово «чакра» в переводе означает колесо. Подобных примеров можно было бы привести очень много.

Другая группа фактов, подтверждающая существование тонкоматериальных планов реальности, связана с современными исследованиямм по нейрофизиологии. Еще в начале XX в. Шеррингтон указал, что, изучая мозг, мы ничего не узнаем об организации высших психических функций. Иными словами, нервная система реализует только вспомогательные функции, связанные с реализацией стереотипных поведенческих актов. Психика, человеческая индивидуальность локализованы за пределами мозга. Это точка зрения была развита и подтверждена современными нейрофизиологическими исследованиями его ученика, лауреата Нобелевской премии Экклса. В книге «Сознание и его мозг», написанной совместно с известным философом К. Поппером, он приводит множество экспериментальных фактов, подтверждающих, что мозг не является местонахождением сознания. Скорее можно утверждать, что тело человека представляет собой что-то вроде биоробота с набором стандартных поведенческих программ, управляемого сознанием через мозг, служащий чем-то вроде приемника и вспомогательного сервисного компьютера.

О том, что сознание не является продуктом деятельности мозга, говорят также и многочисленные свидетельства о феноменах посмертного существования. Это и переживания людей, побывавших в клинической смерти, и так называемые кармические воспоминания о предыдущих жизнях*. Вопрос о том, насколько можно доверять подобным свидетельствам, пока все еще открыт.

В заключение можно сослаться на разнообразные парапсихологические исследования, в которых осуществляется приборная регистрация феноменов тонкоматериального (астрального) мира.

Таким образом, можно с определенными оговорками утверждать, что если существование тонкоматериальной реальности еще и не подтверждено достаточно точно, то в пользу этого свидетельствуют довольно убедительные и разнообразные группы фактов. Рассмотрим теперь, как изменяют-

---

* Подробнее эта проблема освещена в недавно переведенной на русский язык книге: *Калиновский П.* Переход. М., 1991. — *Примеч. автора.*

ся наши представления о реальности и месте человека в ней с учетом тонкоматериальных планов.

Окружающая нас реальность рассматривается как совокупность ряда планов. С одной стороны, *мира плотных форм*, того, что принято было называть материальным миром, а с другой стороны, *мира проницаемых форм*, как его называли в древних эзотерических учениях, тонкоматериального, астрального мира. Важно подчеркнуть, что это разделение в значительной степени условно, это, скорее, просто два подхода к описанию разных аспектов единой объективной реальности.

Мир плотных форм включает объекты, состоящие из вещества, которые мы воспринимаем обычными органами чувств. Объекты этого мира, как правило, имеют четкую границу, обладают определенной формой, размерами и т. д.

В мир проницаемых форм входят различные полевые устойчивые структуры, имеющие вихревую, а возможно, и какую-нибудь другую природу. Они могут быть образованы какими-то известными видами полей либо, быть может, пока не открытыми. Энергии объектов этого мира обычно невелики, в нем преобладает **информационный аспект реальности**. Будучи образованы полями, они не имеют четких границ, проницаемы, могут менять свою форму и размеры. Как правило, мир проницаемых форм не доступен непосредственному восприятию органов чувств человека. В силу его низких энергий и преимущественно информационной природы затруднена и приборная регистрация, хотя уже сейчас существует целый ряд датчиков и приборов для визуализации его объектов.

Мир проницаемых форм становится более доступен нашему восприятию в особых психических состояниях — измененных состояниях сознания, когда обычные органы чувств заторможены или полностью депривированы. В этих случаях, по всей видимости, включаются другие механизмы восприятия реальности — так называемое нелокальное взаимодействие, когда как канал поступления информации, так и орган восприятия и обработки сигналов не локализованы. Такой тип взаимодействия возможен в так называемых диссипативных системах, изучаемых синергетикой информационных процессов.

К этому классу систем принадлежит и наш мозг, который при возникновении измененных состояний сознания превращается одновременно и в орган восприятия и переработки информации. Важно подчеркнуть, что воспринимаемая таким образом информация не структурирована, т. е. не разбита на отдельные элементы, из которых, как из кирпичиков, складывается целостный образ. Знание возникает здесь как единая, нечленимая целостность, о чем свидетельствуют как описания мистиков древности, так и отчеты наших современников, побывавших в подобных состояниях. Это, с одной стороны, дает ни с чем не сравнимую глубину проникновения в получаемое знание, с другой стороны, сильно затрудняет его передачу доступными нам языковыми средствами. Любое словесное описание оказывается лишь жалкой тенью того, что воспринимается человеком в подобном состоянии.

Согласно современным представлениям, организм человека принадлежит к так называемым диссипативным системам, которые могут существовать только при взаимодействии с окружающей средой. Так, скажем, компьютер или телевизор можно просто отключить от сети питания, и он будет безо всякого ущерба для себя ожидать следующего включения. Если же человеку не дать дышать, то уже через очень небольшое время наступают необратимые изменения и смерть. То же самое имеет место с водой и с пищей, только времени будет немного больше. Примерно таким же образом обстоят дела и с информацией. Ограничение информационного потока, поступающего к человеку, может привести к серьезным необратимым изменениям в психике. Заметим здесь, что изоляция от слабых электромагнитных полей, никогда не ощущаемых человеком, не менее отрицательно влияет на живой организм. Так, в экспериментах на крысах такая изоляция приводила к возникновению онкологических заболеваний.

Обобщая особенности нарушений, связанных с изоляцией живых организмов или их сообществ от окружающей среды, можно видеть, что все они приводят к нарастанию хаоса, рассогласования в работе

органов. Естественно предположить, что должны существовать специальные механизмы, противодействующие хаосу, поддерживающие целостность и гармонию сложной системы, будь это отдельное живое существо или зоологическое общество. С глубокой древности их месторасположением считали «тонкоматериальное» или «тонкое» (астральное) тело человека, о существовании которого говорят практически все древние учения.

Описание «анатомии» и функций этого «тонкого» тела совпадает у разных народов. Его органами являются те же самые чакры, или энергетические центры, а своеобразными нервами и сосудами — акупунктурные каналы, связывающие чакры с сердцем, легкими, печенью и т. д. Об аналогичных «тонких» телах, поддерживающих целостность зоологических систем и человеческих сообществ (родов, племен), упоминается и в индийской тантре, и в китайском даосизме. В языческой, шаманской традиции их называли духами-хранителями.

Об одной из основных причин замалчивания существования тонкоматериального мира прекрасно говорит крупнейший исследователь глубинных планов человеческой психики Карл Юнг: «...Здесь перед нами психическая реальность, которая по меньшей мере равноценна физической... В чувстве мы переживаем нечто знакомое, но вещее чаяние ведет нас к неизвестному и сокровенному, к вещам, которые таинственны по своей природе... Они скрыты от человека, а он, из суеверия, буквально «боязни демонов» прячется от них, укрываясь за щит науки и разума. Упорядоченный космос есть его дневная вера, которая призвана уберечь его от ночных страхов хаоса — просвещение из страха перед ночной верой! ...Было бы не лишено интереса исследовать, не окажется ли изобретенное в Новое время отвращение к суеверию и столь же новоевропейское материалистическое просветительство своего рода производным и дальнейшим ответвлением первобытной магии и страха перед духами...»

Перейдем теперь к рассмотрению отражения пространственно-временных взаимосвязей и взаимодействий в астральном мире.

IV. Измененные состояния сознания и параллельные вселенные

Модель физического мира, его пространства и времени, формируемая в психике человека, несет ограничения, связанные как с логикой работы психических процессов, так и культурными традициями общества, его господствующей парадигмой. Поэтому описание геометрии и структуры внутреннего пространства, моделирующего и отображающего внутренний мир в психике человека, представляет собой один из важнейших этапов в понимании внутренней логики и мотивов поступков человека. Подчеркнем, что внутреннее пространство, первоначально возникнув как модель внешнего физического мира, переросло эту функцию, превратившись в арену событий внутренней жизни человека, объясняющих мотивы многих нелогичных, странных и даже диких его поступков. В таких внутренних событиях видна разница между законами, геометрией физического и внутреннего пространства.

В измененных состояниях сознания контакт человека с физическим миром сводится к минимуму, и события разворачиваются в соответствии с логикой работы психических процессов. Поэтому анализ особенностей геометрии пространства мифов, снов, фантазий дает прекрасный материал для выделения закономерностей внутреннего пространства человеческой психики. Анализируя связь субъективных представлений о времени и пространстве с господствующей в обществе технологией, можно заметить, что такая технология и служит основой модели, на которой строится описание внешнего мира. Так, в древнем мире основными технологическими средствами были веретено и гончарный круг. Именно этим можно объяснить то особое значение, которое придавалось круговому движению в философии и космогонии тех эпох. Достаточно вспомнить идею циклического, бесконечно повторяющегося времени в древнеиндийской культуре или античную концепцию вращающегося универсума.

Одним из популярнейших символов ранней европейской культуры стали часы, а позднее — паровой двигатель. Именно эти базовые технологические процессы во многом дали толчок к возникновению кар-

тины мира средневековья и Возрождения. Действительно, концепцию часового механизма с его механическим, заранее детерминированным двигателем нетрудно увидеть в кеплеровской космогонии, аналогии с часовым механизмом использовались не только в описаниях макрокосмоса, но и в моделях микрокосма, например, у Гюйгенса, Декарта, Лейбница.

Наконец, в XX столетии возникла принципиально новая компьютерная метафора. Особенность компьютера в том, что он не содержит движущихся частей, т. е. не является механическим устройством, а имеет чисто геометрическую структуру. Его появление привело к возникновению новой метафоры: «мозг — вычислительная машина». И хотя было бы грубой ошибкой отождествлять психические процессы с работой вычислительной машины и тем более приписывать современным компьютерам функции разума, использование подобных аналогий бывает весьма продуктивно. Являясь по сути первой информационной машиной, созданной человечеством, компьютер позволяет заново осмыслить и понять многие процессы, происходящие в психике человека.

В частности, развитие современных компьютеров может служить прекрасным примером закономерностей формирования внутреннего пространства, моделирующей реальный мир информационной системы. Особенности такого компьютерного пространства тесно связаны с характеристиками логического процесса в вычислительной машине, а именно *дискретностью, конвенциональностью, ограниченностью и изолированностью*. Собственно компьютерное пространство складывается из двух компонентов: физического пространства, в котором находятся электронные элементы и другие детали компьютера как машины, и логического (которое по сути и есть внутреннее пространство), представляющего свойства компьютера как логической сущности, не привязанной к его механическим частям.

Нетрудно увидеть, что компьютерное пространство в огромной степени определяется свойствами памяти вычислительной машины, которая представляет собой декартову систему координат. Поэтому такое внутреннее пространство подобно Ньютонову пространству. В то же время оно всегда конечно, а числа в нем дискретны и телесны. Это сближает его с универсумом древних космогоний, в частности с пифагорейской школой.

Исходя из таких же принципов, можно попытаться реконструировать и внутреннее пространство человеческой психики, которое еще сложнее, еще менее похоже на геометрию ньютонова пространства.

Внутреннее время компьютера также довольно сильно отличается от физического времени. Если ранее собственное время не входило в характеристику орудий труда, измерялось повседневным опытом человека, его событийностью, то время компьютера определяется последовательностью операций в машине, ее внутренними событиями. Темп такого собственного времени может довольно сильно меняться и задается программистом, оно циклично и конечно. Тут также нетрудно увидеть сильные параллели с космогонией древних.

Однако такие компьютерные время и пространство так же линейны, как и в классической физике. Внутреннее психическое пространство и время, однако, этим свойством не обладают.

Остановимся теперь на особенностях геометрии внутреннего мира человека. Как уже говорилось, под внутренним пространством и временем будем понимать геометрическую интерпретацию психики человека как интеллектуальной и логической сущности, освобожденной от его нейрофизиологических частей. Иными словами, это абстрактное пространство, порождаемое логикой и структурой отображения человеческой психикой внешнего физического мира.

Важно отметить, что, несмотря на внешне субъективный характер внутреннего пространства, оно имеет целый ряд закономерностей и инвариантов, свойственных всем людям, независимо от их культурной и национальной принадлежности. Такие инварианты могут проявляться в различных сказках, художественных произведениях, религиозных и культурных традициях, в снах и галлюцинаторных образах. Они связаны не только с пространственно-временными отношениями, но и с сюжетной структурой художественных

произведений, ценностной ориентацией человека.

В настоящее время еще трудно ответить, какие реальные процессы обусловливают существование подобных инвариантов. Вполне возможно, что они представляют собой проявление закономерностей работы нейрофизиологических механизмов головного мозга, его предопределяемой генетически информационной структуры.

Попытаемся теперь выделить основные инварианты внутреннего пространства, обусловливающие его специфику, отличие от физического пространства, непосредственно взаимодействующего с человеком (ньютонова пространства). Во-первых, отметим отсутствие неизменных размеров и даже формы у одного и того же объекта. За считанные мгновения он может вырастать до гигантских размеров и, наоборот, становиться микроскопически малым. При этом громадными обычно представляются объекты, которые доминируют над данным человеком, представляют для него опасность, а маленькими, наоборот, подчиненные ему. При этом смена отношений подчинения/доминирования может приводить к резкому изменению размеров объекта.

Для образов внутреннего пространства характерна также деформация относительных размеров. Какая-то часть может увеличиваться и становиться огромной, в то время как другая резко уменьшается. Такие деформации относительных размеров часто возникают в детских снах, их можно увидеть во многих ритуальных рисунках и скульптурах, отражающих восприятие в измененных состояниях сознания человека.

Таким образом, если в ньютоновом пространстве инвариантными являются размеры и форма объекта (если, конечно, они не деформируются внешними силами), то во внутреннем пространстве неизменны лишь наиболее общие топологические характеристики объектов, а размеры и форма могут трансформироваться в очень широких пределах.

Во-вторых, если обычное физическое пространство однородно и изотропно, то во внутреннем существуют сильнейшие неоднородности и топологические склейки, делающие возможным мгновенный переход из одной геометрии пространства в другую. Иными словами, тут скорее можно говорить о множестве пространств с разной геометрией, соединенных тоннелями и образующих сложную сеть. Такие сложные пространственные структуры в физике используют для описания эффектов квантовой теории гравитации.

При описании перехода из одного такого пространства в другое обычно используют аналогии узкой щели, трещины, колодца, тоннеля. Этот мотив постоянно присутствует в мифах, сказках. Образ узкого тоннеля возникает и при описаниях состояний, переживаемых в момент клинической смерти. Недаром ритуал прохождения через узкий проход, лаз и т. д. присутствует практически во всех древних обрядах инициации. Тут можно провести аналогии и с прохождением новорожденного через родовые пути.

Еще одной характерной особенностью, связанной с переходом через «тоннель» или точку сингулярности, если пользоваться физической терминологией, является инверсия многих качеств объекта, переходящего такую трансформацию. Так, левое может стать правым, наружное — внутренним, женское — мужским. Отсюда свойственное многим древним мифологиям представление, что один и тот же объект может одновременно обладать противоположными качествами (например, андрогин), существовать в нескольких ипостасях, быть частью чего-то большего и одновременно вмещать его в себя. Все это связано, по-видимому, с тем, что один и тот же объект или образ может проецироваться в различные области внутреннего, моделирующего разные аспекты реальности, пространства, обладающие разной структурой и геометрией и даже размерностью. Иными словами, *внутреннее пространство представляет собой сеть из сравнительно независимых областей, соединенных через точки сингулярности*.

Необходимость такого многомодального описания, проявляющаяся в существовании нескольких метафорических языков (областей внутреннего пространства), используемых для отражения одного и того же явления, можно понять с позиций современной математики и кибернетики. Действительно, каждое такое описание, яв-

ляющееся отражением определенных свойств воспринимаемого объекта, представляет его модель. В то же время построение абсолютно полной и непротиворечивой модели принципиально невозможно. Единственная альтернатива — создание семейства моделей, отражающих отдельные аспекты явления и в совокупности дающих более полное описание. Однако каждая из наших моделей только частично совпадает с объектом описания, поэтому подобные модели в чем-то даже могут быть несовместимы. Классическим примером такого «противоречивого» описания является квантово-волновой дуализм, лежащий в основе квантовой механики, согласно которому один и тот же объект одновременно обладает свойствами и волны, и частицы, хотя, казалось бы, одно исключает другое.

В целом инварианты внутреннего пространства носят более общий и универсальный характер, т. е. его отдельные области могут имитировать и законы ньютоновской механики, что и происходит при повседневном, рутинном взаимодействии человека с окружающей средой. Однако при более сложных психических процессах, особенно в экстремальных ситуациях, творчестве, глубоких эмоциональных и духовных переживаниях, вступают в действие более мощные информационные моделирующие механизмы.

Естественно возникает вопрос: какая физическая реальность стоит за множеством миров, воспринимаемых человеком в измененных состояниях сознания? В наши дни, конечно, совершенно очевидна невозможность сведения всего бесконечного разнообразия образов и ощущений в измененных состояниях сознания к галлюцинациям и фантазиям. Однако и в нашем привычном трехмерном мире вещественных форм нет места для этого бесконечного множества пространств.

Какую-то часть опыта в измененных состояниях сознания можно объяснить в терминах тонкоматериального мира, т. е. мира, состоящего из устойчивых полевых структур (вихрей, солитонов и т. д.), находящегося в нашем же трехмерном пространстве и обычно не воспринимаемого нами по тем же причинам, по которым мы не ощущаем радиоволн, радиационных полей, элементарных частиц, буквально пронизывающих и наши тела, и окружающую среду. Однако более глубокие погружения уже не укладываются в подобные объяснения, особенно когда мы сталкиваемся с совершенно иной геометрией пространства и времени. И тут очень плодотворным оказывается представление о **многомерном пространстве и времени**.

Для большей иллюстративности обычно используется аналогия с двухмерным пространством. Живя в трехмерном пространстве, мы можем воспринимать только тени, проекции четырехмерных объектов так же, как на плоскости невозможно увидеть целиком трехмерный объект. Это может рождать массу сложностей и ошибок восприятия. Например, многие боги индийского и тибетского пантеона изображаются с множеством разных лиц, совершенно не похожих друг на друга, одни прекрасные, другие страшные и уродливые. С точки зрения древней эзотерики, боги — многомерные существа, и, являясь в наш мир, они показывают лишь одну из своих проекций, воспринимаемых нами как отдельное тело. И эти различные лики и множество рук символизируют разные проекции — облики одного и того же бога — многомерного существа. Аналогично можно пытаться интерпретировать один из самых сложных образов христианства — триединство Бога-отца, сына и духа.

В четвертом измерении может быть упаковано множество вселенных, столь же и даже более сложных, нежели наша. При этом из одной вселенной можно попасть в другую через гиперпространственный тоннель, который может соединять как очень удаленные точки одной вселенной, так и различные миры. Он может не обладать протяженностью или же восприниматься как длинный лаз, ход, труба.

Проникнуть в параллельную вселенную можно в «тонком» теле, имеющем полевую природу. Подобные путешествия все мы совершаем во время снов, когда наше вещественное тело остается в кровати, а «тонкое» оказывается в одной из близких Вселенных. Иными словами, гораздо легче найти тоннель, прозрачный для «тонкого» тела, именно поэтому подавляющее число подобных путешествий совершается в измененных состояниях созна-

ния, когда «тонкое» и вещественное тела разделяются и оказываются в разных мирах.

С древности было известно, что в определенных географических местах переходы в параллельные миры осуществить легче. Причем каждое такое место тяготеет к тому или иному типу миров. Однако было бы неверно считать, что каждое место может использоваться только для перехода лишь в один из миров. На самом деле существует целый спектр возможностей, но с разной вероятностью, т. е. в одни вселенные попасть через данное место легче, в другие сложнее. Но, используя определенные техники настройки, можно изменить эти вероятности и в нужной вселенной. Таким образом, каждое место перехода можно характеризовать распределением вероятностей попадания в тот или иной мир. Распределение таких вероятностей и образует индивидуальный портрет каждой области нашего пространства, через которую может осуществляться гиперпространственный переход.

Современные данные и громадный эмпирический материал, накопленный древними цивилизациями, позволяют предположить, что пространство в области гиперпространственного перехода обладает особой структурой. Удачным аппаратом для ее описания могла бы стать теория фракталов, описывающая так называемые фрактальные кривые — линии, заполняющие все точки плоскости (построение которых рекурсивно), примерами которых являются кривая Коха, ковер Серпинского и т. п. Современные исследования показали, что многие реальные физические объекты обладают свойствами фракталов. Это облака, изрезанная береговая линия, поверхность гор, скал, внутреннее пространство пещер, различные ворсистые поверхности и т. д.

По всей видимости, совершение гиперпространственного перехода в «тонком» теле, возможно в любой точке нашего пространства. Естественно, где-то это сделать проще, где-то сложнее. Однако, используя специальные приемы, этот процесс можно существенно облегчить. Подобные приемы были известны в самых различных древних цивилизациях, и все они очень похожи и распадаются на три основных класса.

Во-первых, это подготовка пространства в месте предполагаемого перехода (упомянем здесь и специальные архитектурные сооружения, и биорезонаторы, и геометрические фигуры типа янтр и мандал, а также специальные магические ритуалы, предназначенные для открытия «двери» в параллельную вселенную).

Во-вторых, это формирование специального психоэнергетического состояния, в котором возможен переход в «тонком» теле. Это обычно разновидность измененного состояния сознания, особого транса. Тут также используется широкий спектр методов, начиная от разнообразных наркотических веществ и кончая дыхательными и психоэнергетическими приемами, специальной музыкой, танцами. В наши дни сюда добавился широкий спектр современных технических средств — это депривационная ванна, различные приборы для зрительной, слуховой, электромагнитной стимуляции. На этом этапе огромную роль играют структуры «тонкого» тела — чакры и акупунктурные каналы.

Наконец, в-третьих, это настройка на переход в определенную параллельную вселенную, в ее определенное место и время. Понятно, что этот, третий, этап очень важен.

Можно сказать, что если первые приемы открывают «дверь», то второй и третий классы техник создают «ключ» от нее. Естественно, что использование всех подобных методик предполагает достаточно очищенное гармонизированное подсознание, поэтому подобным путешествиям в древности обычно предшествовал сложный и длительный процесс обучения и психотрансформации с последующим посвящением.

Настоящая работа выполнена по заказу Министерства обороны Российской Федерации.

Авторы выражают благодарность академикам Российской академии естественных наук (РАЕН) А. Ю. Савину и В. В. Щенникову за плодотворное обсуждение при написании данной статьи.

Л. А. Залогова,
кандидат физико-математических наук, Пермский государственный университет, Лаборатория информатизации образования Пермской области
И. Г. Семакин,
кандидат физико-математических наук, Пермский государственный университет, Лаборатория информатизации образования Пермской области

БАЗОВЫЙ КУРС ОИВТ: «ПЕРМСКАЯ ВЕРСИЯ» (часть 4)

Тема 7. Компьютер рисует

В этом разделе изучается применение ЭВМ для получения графических изображений. Под словами «графическое изображение» подразумеваются самые разнообразные рисунки, картинки, чертежи, графики и пр., которые получаются на экране компьютера, а также могут быть выведены на печать. На экране рисунки могут быть статическими (неподвижными) или динамическими (движущимися). В последнее время машинная графика выделилась в самостоятельный раздел информатики с многочисленными приложениями. Средствами машинной графики создается не только печатная продукция, но и рекламные ролики на телевидении, мультфильмы.

Все современные персональные компьютеры снабжены аппаратными и программными средствами для получения графических изображений. Такие средства имеются и на всех типовых моделях школьных компьютеров. При изучении темы используются несложные графические редакторы.

Кодирование изображения

Представьте себе рисунок, на который наложена сетка с квадратными ячейками. В каждую ячейку попадает маленький фрагмент рисунка. Если брать сетку все гуще (ячейки — все меньше), то в конце концов в каждой ячейке окажется одноцветная точка. Тогда весь рисунок представляется как совокупность таких точек (мозаика из точек).

Идея представления изображения как совокупности точек используется в машинной графике. О получении изображения символов из точек говорилось в теме «Компьютер пишет». Но там речь шла о получении стандартных изображений символов в фиксированных позициях экрана. В графическом режиме можно получать любые изображения, управляя состоянием любой точки экрана.

Точечный элемент экрана компьютера называется п и к с е л о м (от английского pixel — picture element). Совокупность пикселов на экране образует графическую сетку. Чем гуще эта сетка, тем лучше будет качество изображения.

Информация о графическом изображении хранится в специальном разделе оперативной памяти компьютера, который называется видеопамятью. В видеопамяти содержится информация о состоянии каждого пиксела экрана. Если каждый пиксел может принимать только два состояния: светится — не светится (белый — черный), то для кодировки достаточно одного бита памяти (1 — белый, 0 — черный). Если нужно кодировать большее количество состояний (различную яркость свечения или различные цвета), то одного бита на пиксел недостаточно.

Обсудим подробнее вопрос о том, как кодируются и получаются цветные графические изображения на экране.

Цвет точки на экране формируется из трех основных цветов: красного, синего, зеленого. Различные цвета получаются в результате наложения световых пятен, возникающих под действием лучей трех электронных пушек. Цветовая палитра увеличивается, если имеется возможность управлять интенсивностью (яркостью) основных цветов.

Расшифровка номеров цветов, применяемых на IBM PC, показана в табл. 1.

Таблица 1

И	К	З	С	Номер цвета	Цвет
0	0	0	0	0	черный
0	0	0	1	1	синий
0	0	1	0	2	зеленый
0	0	1	1	3	голубой
0	1	0	0	4	красный
0	1	0	1	5	розовый
0	1	1	0	6	коричневый
0	1	1	1	7	серый (или белый)
1	0	0	0	8	темно-серый
1	0	0	1	9	ярко-синий
1	0	1	0	10	ярко-зеленый
1	0	1	1	11	ярко-голубой
1	1	0	0	12	ярко-красный
1	1	0	0	13	ярко-розовый
1	1	1	0	14	ярко-желтый
1	1	1	1	15	ярко-белый

Биты: И — интенсивность,
К — красный,
З — зеленый,
С — синий.

Из таблицы видно, что различные комбинации красного, зеленого и синего дают 8 главных и 8 фоновых (с пониженной яркостью) цветов. В шестнадцатицветных режимах можно использовать любые цвета от 0 до 15 (4 бита видеопамяти на пиксел); в восьмицветных — от 0 до 7, т. е. все цвета без бита интенсивности (3 бита на пиксел). В четырехцветных режимах нужно выбрать 4 цвета из шестнадцати (2 бита на пиксел). Палитра может включать 64 цвета. В этом случае также используются три составляющие (красный, зеленый, синий). Однако каждый цвет теперь кодируется не одним, а двумя независимыми сигналами — основным (большей интенсивности) и дополнительным (меньшей интенсивности). Обозначим прописными буквами (К, З, С) менее яркие сигналы, а строчными (к, з, с) — более яркие. Два независимых сигнала дают четыре возможных уровня яркости каждого цвета. Например, для красного: .. (нет красного), Кк (самый яркий красный), ..К , . .к . Возможные уровни яркости основных цветов приведены в табл. 2.

Таблица 2

Красный	Синий	Зеленый
Кк	Сс	Зз
К..	С..	З..
к..	с..	з..
..	..	..

Все возможные комбинации К, к, С, с, З, з... дают 64 цвета.

Из всего сказанного следует: необходимый размер видеопамяти зависит от размера графической сетки (количества пикселов) и от количества цветов, которые требуется получать на экране.

Возьмем для примера память объемом 16 Кбайт. Этот объем достаточен для отображения 640×200 точек, если с каждым пикселом связать лишь один бит видеопамяти (640×200×1=128 000 бит). Каждый бит может иметь нулевое или единичное значение. Нулевое значение бита — это сигнал об отсутствии светимости соответствующего пиксела (черный цвет). И наоборот, единичное значение — светимость каким-либо заранее выбранным цветом. Таким образом, объем видеопамяти 16 Кбайт позволяет получить на экране

размером 640×200 точек только двухцветное изображение. Для получения четырехцветного изображения на каждый пиксел необходимо выделить два бита видеопамяти. Пара битов принимает значения 00, 01, 10, 11. Тогда встает вопрос: какие четыре цвета выбрать из шестнадцати доступных?

Возможны следующие решения:

1) все цвета образуются сочетанием красного и зеленого (синий луч не испускается). Если пара битов установлена в состояние 01, точка загорается красным цветом, 10 — зеленым, 11 — их смесью, т. е. коричневым, 00 — черным;

2) синяя составляющая всегда присутствует. Состоянию 01 соответствует смесь красного и синего, 10 — зеленого и синего. Если же оба бита включены, получается смесь всех трех цветов, воспринимаемая как белый.

Поскольку объем видеопамяти всегда ограничен (в нашем примере 16 Кбайт), количество точек экрана и возможных цветов каждой точки также ограничено. Причем эти величины взаимно связаны. Если в два раза увеличить количество битов на пиксел, то в два раза уменьшится количество точек по одной из осей экрана при том же объеме видеопамяти (320×200×2=128 000 бит).

В большинстве ЭВМ существует несколько режимов работы. Например, машины серии IBM PC, которые оснащены графическим адаптером Color Graphic Adapter (адаптер цветной графики), могут работать в следующих режимах:

320×200 пикселов, 4 цвета;

640×200 пикселов, 2 цвета.

Монитор типа VGA реализует такой графический режим:

640×480 пикселов, 16 цветов.

Персональная ЭВМ Atari 1040ST (США) имеет графический дисплей, работающий в следующих режимах:

320×200 пикселов, 16 цветов;

640×200 пикселов, 4 цвета;

640×400 пикселов, 2 цвета.

Персональная ЭВМ КУВТ-86 работает в двух режимах:

511×238 пикселов, 2 цвета;

256×238 пикселов, 4 цвета.

Графические редакторы

Для получения графических изображений на ЭВМ используется специальное программное обеспечение — графический редактор (ГР) [1], [2]. Пользователя графическим редактором будем называть художником, его цель — создание рисунка.

Рисовать изображение можно в режиме ручной прорисовки или из базовых инструментов (примитивов). Выбор в качестве инструмента «кисточки» позволяет наносить изображения на «холст» от руки. В этом случае художник использует курсор, управляемый клавишами, или манипулятором типа «мышь». Очень трудно с помощью «мыши» от руки провести прямую линию. Используя в качестве инструмента «линейку», достаточно просто соединить прямой любые две точки рабочего поля.

Можно без труда нарисовать окружность, квадрат или многоугольник произвольной формы. Для этого нужно выбрать в таблице инструментов графический примитив и установить курсор в соответствующую точку рабочего поля. Для прямоугольника — это левый верхний угол, для круга и эллипса — центр. При этом «художник» может подобрать желаемые размеры примитива.

При помощи графического редактора художник должен иметь возможность строить изображения путем компоновки их из других, ранее созданных изображений, объединяя их с текстом и изменяя цвета. Поэтому в графическом редакторе должны быть реализованы функции, позволяющие

- «вырезать», «склеивать» и «стирать» произвольные части изображения;
- применять для рисования произвольные «краски» и «кисти»;
- запоминать рисунки на внешних носителях, осуществлять их поиск и воспроизведение;
- увеличивать фрагмент изображения для проработки мелких деталей;
- добавлять к рисункам текст и таким образом создавать красочные объявления, рекламные плакаты, визитные карточки и т. д.

Графический редактор позволяет также масштабировать (изменять размер)

Рис. 1

Обозначения: **ЦП – центральный процессор;**
ОП – оперативная память.

изображения, выполнять его перемещение и поворот.

Так же как текстовый редактор, графический редактор рассматриваем как исполнителя команд пользователя-«художника». Художнику известен алгоритм получения нужного изображения. Работая по этому алгоритму с ГР, он строит рисунок.

Описание графического редактора как исполнителя будем вести по прежней методической схеме.

Аппаратные средства

При получении графического изображения на ЭВМ задействованы все основные аппаратные компоненты компьютера. Аппаратной основой машинной графики является графическая система, схема которой представлена на рис. 1.

Графический адаптер представляет собой единство двух компонент: видеопамяти и дисплейного процессора.

Про назначение видеопамяти уже было сказано. Ее функция — хранить видеоинформацию.

Функция дисплейного процессора — выводить содержимое видеопамяти на экран. Если изображение на экране постоянно не подновлять, то оно гаснет (за время порядка нескольких миллисекунд). Таким образом, изображение должно выводиться на экран с такой частотой, чтобы глаз не успевал заметить угасание картинки. Дисплейный процессор непрерывно просматривает видеопамять и выводит ее содержимое на экран 50—60 раз в секунду.

Существенным моментом является то, что видеопамять находится под непосредственным управлением центрального процессора. Это такая же оперативная память, как и вся остальная. Центральный процессор может читать и изменять ее содержимое. Таким образом, с видеопамятью одновременно работают сразу два процессора — центральный и дисплейный (последний — только на чтение).

Графический дисплей обеспечивает отображение графической информации на экране электронно-лучевой трубки. В настоящее время широкое распространение получили растровые дисплеи. Экран растрового дисплея разбит на фиксированное число точек, которые образуют матрицу («растр») из фиксированного числа строк и столбцов.

. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .

Рис. 2

Слово «растр» восходит к латинскому rastrum — «грабли», «мотыга». Растром обычно называют чередование прозрачных и непрозрачных полос по сходству со следом граблей, имеющим вид параллельных борозд. Растровые дисплеи работают в прямоугольной декартовой системе координат. Каждый пиксел характеризуется координатами — парой чисел (X, Y). Первое число X задает расстояние от начала координат до заданной точки экрана по горизонтали (в пикселах), второе число Y — по вертикали. В большинстве ЭВМ требуется, чтобы эти координаты изменялись слева направо и сверху вниз. Это означает, что экран дисплея связан с системой координат, начало которой находится в левом верхнем углу экрана (рис. 3).

Координата X при движении слева направо принимает целочисленные значения 0, 1, 2, ... до некоторого максимального значения XM. Координата Y увеличивается сверху вниз, принимая значения 0, 1, 2, ... YM . Точка, расположенная в верхнем левом углу, имеет координаты (0, 0); а точка в правом нижнем углу — координаты (XM, YM). Такая система координат используется в ЭВМ фирмы IBM, а также на персональном компьютере КОРВЕТ. В некоторых ЭВМ применяется система координат с началом в левом нижнем углу экрана.

Величины, характеризующие ширину и высоту экрана (в пикселах), т. е. XM и YM, в разных системах могут меняться от десятков до нескольких сотен и тысяч. Чем больше XM и YM, тем выше качество изображения, так как каждая точка будет занимать меньшую область на экране. Количество пикселов по горизонтали и вертикали (XM и YM) называется разрешающей способностью. Горизонтальные и вертикальные линии выглядят гладкими при любой разрешающей способности системы. Однако для диагональных линий чем ниже разрешающая способность системы, тем больше прямая похожа на ступеньки лестницы.

Цветные мониторы, работающие по принципу смешения трех основных цветов, называют RGB-мониторами (Red, Green, Blue).

При работе с графическим редактором в качестве устройств ввода данных и команд используются клавиатура, а также манипуляторы «мышь» или «джойстик» (рычаг).

«Мышь» — коробочка, легко умещающаяся в руке, с одной или двумя кнопками наверху. На дне коробочки располагается два колесика. Для того чтобы рисовать «мышью», достаточно катать ее по ровной поверхности. При этом смотреть нужно не на «мышь», а на экран дисплея, где траектория движения курсора повторяет траекторию движения «мыши».

Джойстик напоминает ручку управления автомобилем или самолетом. Наклоняя рычаг в нужную сторону, можно добиться плавного перемещения курсора

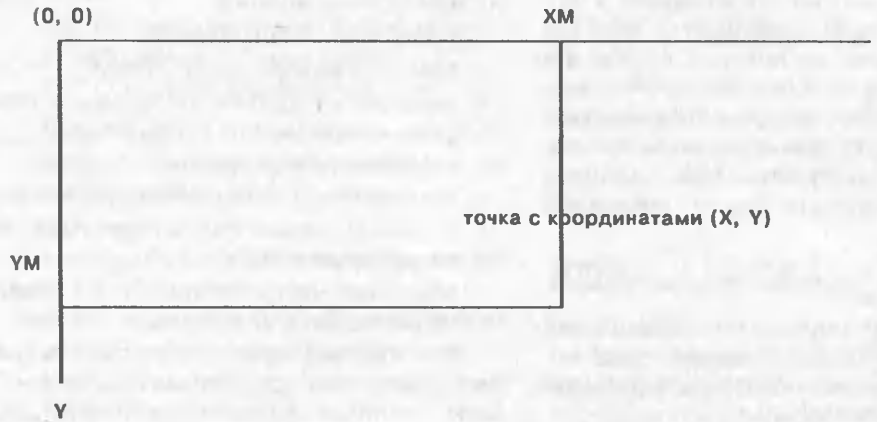


Рис. 3

на экране. Причем чем больше наклон рычага, тем быстрее перемещается курсор.

Некоторые ГР позволяют напечатать рисунок на бумаге с помощью принтера.

Принтер печатает картинку по строкам и никогда не возвращается в ту часть картинки, которая уже напечатана. Устройства цветной печати используют вместо красящей ленты баллончики с чернилами разных цветов, капельки которых «выстреливаются» пишущим узлом в определенные точки растра.

В комплекте аппаратуры профессиональных моделей персональных компьютеров могут использоваться сканеры. Сканер позволяет вводить в ЭВМ изображения с репродукций. Затем введенное изображение может подвергаться обработке с помощью графического редактора.

Программные средства ГР

Среда графического редактора.

Пользовательский интерфейс большинства графических редакторов организуется следующим образом. С левой стороны экрана располагается набор пиктограмм (условных рисунков) с изображением инструментов, которыми можно пользоваться в процессе редактирования изображений. В нижней части экрана — палитра, из которой «художник» выбирает краски требуемого цвета. Оставшаяся часть экрана представляет собой пустой «холст» (рабочее поле). Над рабочим полем находится меню, позволяющее изменять режимы работы ГР. На левом краю палитры выводится квадрат, окрашенный в фоновый цвет. В нем помещаются еще два квадрата, верхний из которых окрашен в первый рабочий цвет, а нижний — во второй рабочий цвет. В левом нижнем углу экрана выводится калибровочная шкала, которая позволяет устанавливать ширину рабочего инструмента (кисти, резинки и т. д.).

Режимы работы графического редактора.

Режимы ГР определяют возможные действия художника, а также команды, которые художник может отдавать редактору в данном режиме.

Режимы работы:

1. Режим работы с рисунком (рисование). В этом режиме на рабочем поле находится изображение инструмента. Художник наносит рисунок, редактирует его, манипулирует его фрагментами.

2. Режим выбора и настройки инструмента. Курсор-указатель находится в поле экрана с изображениями инструментов (меню инструментов). Кроме того, с помощью меню можно настроить инструмент на определенный тип и ширину линии, орнамент закраски.

3. Режим выбора рабочих цветов. Курсор находится в поле экрана с изображением цветовой палитры. Здесь можно установить цвет фона, цвет рисунка. Некоторые ГР дают возможность пользователю изменять палитру.

4. Режим работы с внешними устройствами. В этом режиме можно выполнять команды записи рисунка на диск, считывания рисунка с диска, вывода рисунка на печать. Графические редакторы на профессиональных ПК могут работать со сканером, используя его для ввода изображения с репродукций.

Система команд графического редактора.

В каждом из перечисленных выше режимов художник может работать с определенным набором команд ГР, совокупность которых и составляет СКИ графического редактора. В различных графических редакторах на разных компьютерах системы команд могут существенно различаться. Во всех вариантах характерно использование принципа меню для выбора и инициализации команд.

В систему команд входят:

- команды выбора инструмента;
- команды настройки инструмента (ширина линий, шрифт букв);
- команды выбора цветов;
- команды масштабирования рисунка;
- команды ввода/вывода рисунка на внешние устройства.

Меню команд представляются в форме пиктограмм, а также в текстовой форме.

Проводя обучение на конкретном графическом редакторе, учитель должен сообщить ученикам конкретную систему команд, указав способы их инициализации.

Данные графического редактора

(графическая информация) — рисунки, фрагменты рисунков.

Функция пользователя (художника) состоит в управлении редактором по определенному алгоритму с целью получения нужного рисунка. Алгоритм составляется из следующих действий художника: 1) инициализация команд из системы команд графического редактора; 2) ввод и редактирование рисунка (рисование линии; стирание фрагмента изображения с помощью резинки; раскраска картинки пульверизатором и валиком; выделение фрагмента рисунка произвольной многоугольной формы; копирование фрагмента изображения; перемещение фрагмента изображения; построение изогнутой линии; построение многоугольников; построение кругов и эллипсов; оформление рисунка текстом; редактирование мелких деталей рисунка и др.).

Вот несколько примеров алгоритмов работы с графическим редактором.

Пример 1

Написать алгоритм рисования линии с помощью линейки, используя графический редактор PAINTBRUSH 4.0 [3].

Алгоритм:

1. Выбрать ширину линии.
2. Выбрать цвет линии.
3. Выбрать в таблице инструментов линейку.
4. Нарисовать линию (действие состоит из 4 шагов).

4.1. Установить курсор на место первой точки.

4.2. Нажать левую клавишу «мыши».

4.3. Провести прямую линию.

4.4. Отпустить клавишу «мыши».

Пример 2

Стереть фрагмент изображения с помощью резинки.

Алгоритм:

1. Выбрать в таблице инструментов резинку.
2. Выбрать ширину резинки.
3. Стереть фрагмент изображения.

3.1. Установить курсор на рабочее поле.

3.1. Нажать левую клавишу «мыши».

3.2. Не отпуская клавиши, двигать курсор по экрану (резинка удаляет все, оставляя только фоновый цвет).

Пример 3.

Выделить фрагмент рисунка N-угольной формы.

Алгоритм:

1. Выбрать в таблице инструментов ножницы.
2. Выделить фрагмент изображения.

2.1. Поместить курсор на рабочее поле.

2.2. Нажать любую клавишу «мыши».

2.3. Повторить (N−1) раз.

начало

2.3.1. Переместить курсор на место следующего угла многоугольника.

2.3.2. Нажать ту же самую клавишу «мыши».

конец

2.4. Нажать любую клавишу «мыши» два раза подряд (для замыкания контура).

Пример 4

Переместить выделенный фрагмент изображения на новое место.

Алгоритм:

1. Поместить курсор внутрь выделенного фрагмента.
2. Нажать клавишу «мыши».
3. Переместить курсор на новое место.
4. Отпустить клавишу «мыши».

Пример 5

Вывести картинку в файл на диск.

Алгоритм:

1. Выбрать в меню команду FILE (при этом появится подменю команды).
2. Выбрать в подменю команду SAVE AS (при этом появится поле для ввода имени файла).
3. Ввести имя файла, в который должна быть записана картинка (PAINTBRUSH автоматически добавит к имени файла расширение .PCX и запишет его в текущий каталог).

Планирование уроков

Урок 1

Теоретическая часть. Машинная графика как отрасль информатики. Принципы хранения графической информации в памяти ЭВМ. Экран дисплея в графическом режиме, понятие пиксела. Видеопа-

мять, дисплейный процессор. Формирование цветовой гаммы на экране. В начале или в конце урока (на усмотрение учителя) показать демонстрационные рисунки, ролики, полученные средствами машинной графики.

Урок 2

Теоретическая часть. Назначение графического редактора (ГР). Возможности ГР. Показать (на плакате или на демонстрационном мониторе), как выглядит экран в процессе работы с ГР. Среда ГР. Режимы работы ГР. Система команд ГР. Данные ГР. Примеры алгоритмов работы с графическим редактором.

Практическая часть. Загрузить ГР на рабочие места учеников. После входа в ГР рабочим инструментом является кисть. Порисовать с помощью кисти, выбирая различную ширину и цвет кисти. Рассмотреть использование резинки. Рассмотреть использование линейки, прямоугольника, эллипса, ломаной.

Урок 3

Практическая часть. Изобразить замкнутый контур и выбрать в таблице инструментов валик. Закрасить валиком замкнутый контур. Использование пульверизатора и пипетки. Выделить фрагмент рисунка с помощью ножниц или рамки. Скопировать выделенный фрагмент на новое место. Запомнить рисунок на диске и выйти из ГР. Снова войти в ГР и считать изображение с диска. Показать приемы проработки мелких деталей рисунка путем их увеличения.

Выдать ученикам индивидуальные задания на получение «художественных» картинок, предварительно подготовив раздаточный материал. Самостоятельная работа учеников над индивидуальными заданиями. Полученные рисунки (завершенные или незавершенные) сохраняются во внешней памяти.

Урок 4

Практическая часть. Научить учеников оформлению рисунка текстом (выбор в качестве инструмента пишущей машинки, выбор шрифта, оформления символов). Создать общий чертеж с использованием графических примитивов, оформленный текстом. Выдать, используя раздаточный материал, индивидуальные задания на получение чертежа или схемы с поясняющими текстами, формулами, таблицами (рисунки можно взять из учебников математики, черчения, физики и др.).

Работа над индивидуальными заданиями.

Урок 5

Зачетный урок. Завершение работы над индивидуальными заданиями и их сдача. Контрольный опрос по теоретической части темы в форме письменных ответов на контрольные вопросы или тестов.

Контрольные вопросы и задания

1. В чем суть точечного принципа представления изображения?
2. Что такое **пиксел**?
3. Для чего нужна видеопамять?
4. Из каких трех цветов получаются все остальные цвета?
5. Сколько различных цветов можно закодировать, если в видеопамяти выделяется на 1 пиксел: 1 бит; 2 бита; 3 бита; 4 бита?
6. Какова роль бита интенсивности?
7. От чего зависит необходимый объем видеопамяти?
8. Каким должен быть объем видеопамяти, если графический дисплей работает в режиме 640×480 пикселов, 16 цветов?
9. Какие устройства входят в состав графического адаптера?
10. Что делает графический процессор?
11. С какой частотой выводится содержимое видеопамяти на экран?
12. Какие устройства используются для ввода и вывода графического изображения?
13. Как устроен экран растрового дисплея?
14. Что такое графический редактор?
15. Какие основные виды работ с изображением можно выполнить с помощью графического редактора?
16. Перечислить основные режимы работы графического редактора.
17. Как отдаются команды графическому редактору? Что такое пиктограмма?
18. Перечислить основные типы команд ГР.

Задания для индивидуальных работ

1. Для первого индивидуального задания можно использовать несложные рисунки из детских книг, с открыток, красочные рисунки из школьных учебников. Ученикам можно предложить самим придумать и нарисовать красочное поздравление с праздником или с днем рождения.

2. Для второго индивидуального задания полезно выбрать учебно-иллюстративный материал. Это задание можно трактовать как изготовление наглядного пособия по определенному школьному предмету. Рисунок должен содержать чертеж и текст или формулы. Например, можно нарисовать электрическую цепь и формулы вычисления ее характеристик (сопротивление, ток, напряжение).

Таблица 3

Основные функции графического редактора

	1	2	3	4	5	6
Создание рисунка						
режим ручной прорисовки	+	+	+	+	+	+
с использованием инструментов	+	+	+	–	+	+
Манипулирование рисунком						
выделение фрагментов	+	+	+	+	+	+
проработка мелких деталей	+	+	+	+	+	+
копирование на новое место экрана	+	+	+	+	+	+
закраска отдельных частей рисунка ровным слоем или узором	+	+	+	+	–	+
масштабирование	+	+	+	+	+	+
перемещение	+	+	+	+	+	+
поворот	+	+	+	+	–	–
Оформление рисунка текстом						
выбор шрифта	+	+	+	–	+	+
оформление символов (курсивное начертание, оттенение, подчеркивание и т. д.)	+	–	–	–	+	+
Работа с цветом						
создание своей палитры	+	+	+	–	–	–
создание своего узора для закраски	+	–	–	–	–	–
Работа с ВУ						
запись рисунка на диск в стандартном формате (PCX, TIF и др.)	+	+	+	+	+	+
чтение с диска	+	+	+	+	+	+
печать рисунка	+	–	–	+	–	–
сканирование изображения	+	–	–	–	–	–

Обозначения:

1 – графический редактор PAINTBRUSH IBM PC
2 – графический редактор PAINT КУВТ-86
3 – графический редактор PAINT КОРВЕТ
4 – редактор спрайтов КОРВЕТ
5 – графический редактор GREDIT КУВТ-86
6 – графический редактор PAINT УКНЦ

Литература

1. *Александров В. В., Шнейдеров В. С.* Рисунок, чертеж, картина на ЭВМ. Л.: Машиностроение, 1988.

2. Персональный компьютер: рабочее место профессионала. М.: Наука, 1989.

3. Графический редактор Paintbrush 4.0. Обнинск, 1993.

А. Х. Шень,
научный сотрудник Института проблем передачи информации РАН

ПРОГРАММИРОВАНИЕ: ТЕОРЕМЫ И ЗАДАЧИ*

Глава 2. Порождение комбинаторных объектов

Здесь собраны задачи, в которых требуется получить один за другим все элементы некоторого множества.

2.1. Размещения с повторениями

2.1.1. Напечатать все последовательности длины k из чисел 1...n.

Решение. Будем печатать последовательности в *лексико-графическом порядке* (последовательность a предшествует последовательности b, если для некоторого s их начальные отрезки длины s равны, а (s+1)-й член последовательности a меньше). Первой будет последовательность ⟨1, 1, ..., 1⟩, последней — последовательность ⟨n, n, ..., n⟩. Будем хранить последнюю напечатанную последовательность в массиве x[1]...x[k].

```
...x[1]...x[k] положить равным 1
...напечатать x
...last[1]...last[k] положить равным n
while x<>last do begin
| ...x:=следующая за x последовательность
| ...напечатать x
end;
```

Опишем, как можно перейти от x к следующей последовательности. Согласно определению, у следующей последовательности первые s членов должны быть такими же, а (s+1)-й — больше. Это возможно, если x[s+1] было меньше n. Среди таких s нужно выбрать наибольшее (иначе полученная последовательность не будет непосредственно следующей). Соответствующее x[s+1] нужно увеличить на 1. Итак, надо, двигаясь с конца последовательности, найти самый правый член, меньший n (он найдется, так как по предположению x<>last), увеличить его на 1, а идущие за ним члены положить равными 1.

```
p:=k;
while not (x[p]<n) do begin
| p:=p-1;
end;
{x[p]<n, x[p+1]=...=x[k]=n}
x[p]:=x[p]+1;
for i:=p+1 to k do begin
| x[i]:=1;
end;
```

Замечание. Если членами последовательности считать числа не от 1 до n, а от 0 до n−1, то переход к следующему соответствует прибавлению 1 в n-ичной системе счисления.

2.1.2. В предложенном алгоритме используется сравнение двух массивов x<>last. Устранить его, добавив булевскую переменную l и включив в инвариант соотношение l <=> (последовательность x — последняя).

2.1.3. Напечатать все подмножества множества {1...k}.

Решение. Подмножества находятся во взаимно однозначном соответствии с последовательностями нулей и единиц длины k.

2.1.4. Напечатать все последовательности из k положительных целых чисел, у которых i-й член не превосходит i.

2.2. Перестановки

2.2.1. Напечатать все перестановки чисел 1...n (т. е. последовательности длины n, в которые каждое из чисел 1...n входит по одному разу).

Решение. Перестановки будем хранить в массиве x[1], ..., x[n] и печатать в лексико-графическом порядке. (Первой при этом будет перестановка ⟨1 2...n⟩, последней — ⟨n...2 1⟩.) Для составления алгоритма перехода к следующей перестановке зададимся вопросом: в каком случае k-й член перестановки можно увеличить, не меняя предыдущих? Ответ: если он меньше какого-либо из следующих членов (членов с номерами больше k). Мы должны найти наибольшее k, при котором это так, т. е. такое k, что x[k]<x[k+1]>...>x[n]. После

* Окончание. Начало см.: Информатика и образование. 1994. № 4.

этого x[k] нужно увеличить минимальным возможным способом, т. е. найти среди x[k+1], ..., x[n] наименьшее число, большее его. Поменяв x[k] с ним, остается расположить числа с номерами k+1, ..., n так, чтобы перестановка была наименьшей, т. е. в возрастающем порядке. Это облегчается тем, что они уже расположены в убывающем порядке.

Алгоритм перехода к следующей перестановке:

```
{<x[1],...,x[n−1], x[n]> <> <n,...,2, 1>}
k:=n−1;
{последовательность справа от k —
убывающая: x[k+1]>...>[n]}
while x[k]>x[k+1] do begin
| k:=k−1;
end;
{x[k]<x[k+1]> ... >x[n]}
t:=k+1;
{t<=n, x[k+1]>...>x[t]>x[k]}
while (t<n) and (x[t+1]>x[k]) do begin
| t:=t+1;
end;
{x[k+1]>...>x[t]>x[k]>x[t+1]>...>x[n]}
... обменять x[k] и x[t]
{x[k+1]>...>x[n]}
... переставить участок x[k+1] ... x[n]
в обратном порядке
```

Замечание. Программа имеет знакомый дефект: если t=n, то x[t+1] не определено.

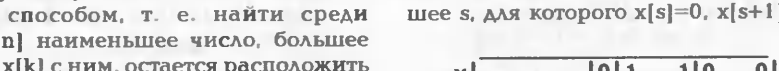
2.2.2. Модифицировать алгоритм перехода к следующей перестановке так, чтобы он сам проверял, не является ли данная перестановка последней.

2.3. Подмножества

2.3.1. Перечислить все k-элементные подмножества множества {1...n}.

Решение. Будем представлять каждое подмножество последовательностью x[1]...x[n] нулей и единиц длины n, в которой ровно k единиц. Такие последовательности упорядочим лексико-графически (см. выше). Очевидный способ решения задачи — перебирать все последовательности, как раньше, а затем отбирать среди них те, у которых k единиц, — мы отбросим, считая его неэкономичным (число последовательностей с k единицами может быть много меньше числа всех последовательностей). Будем искать такой алгоритм, чтобы получение очередной последовательности требовало порядка n действий.

В каком случае s-й член последовательности можно увеличить, не меняя предыдущие? Если x[s] меняется с 0 на 1, то для сохранения общего числа единиц нужно справа от x[s] заменить 1 на 0. Таким образом, x[s] — первый справа нуль, за которым стоят единицы. Легко видеть, что x[s+1]=1 (иначе x[s] не первый). Таким образом, надо искать наибольшее s, для которого x[s]=0, x[s+1]=1:



За x[s+1] могут идти еще несколько единиц, а после них несколько нулей. Заменив x[s] на 1, надо выбрать идущие за ним члены так, чтобы последовательность была бы минимальна с точки зрения нашего порядка, т. е. чтобы сначала шли нули, а потом единицы. Вот что получается:

первая последовательность

0...01...1 (n−k нулей, k единиц),

последняя последовательность

1...10...0 (k единиц, n−k нулей).

Алгоритм перехода к следующей за x[1]...x[n] последовательности (предполагаем, что она есть):

```
s:=n−1;
while not ((x[s]=0) and (x[s+1]=1)) do begin
| s:=s−1;
end;
{s — член, подлежащий изменению с 0 на 1}
num:=0;
for k:=s to n do begin
| num:=num+x[k];
end;
{num — число единиц на участке
x[s]...x[n], число нулей равно
(длина−число единиц), т. е. (n−s+1) — num}
x[s]:=1;
for k:=s+1 to n−num+1 do begin
| x[k]:=0;
end;
for k:=n−num+2 to n do begin
| x[k]:=1;
end;
```

Другой способ представления подмножеств — это перечисление их элементов. Чтобы каждое подмножество имело ровно одно представление, договоримся перечислять элементы в возрастающем порядке. Приходим к такой задаче.

2.3.2. Перечислить все возрастающие последовательности длины k из чисел 1...n в лексикографическом порядке. (Пример: при n=5, k=2 получаем 12 13 14 15 23 24 25 34 35 45.)

Решение. Минимальной будет последовательность 1, 2, ..., k; максимальной — (n−k+1), ..., (n−1), n. В каком случае s-й член последовательности можно увеличить? Ответ: если он меньше n−k+s. После увеличения s-го элемента все следующие должны возрастать с шагом 1. Получаем такой алгоритм перехода к следующему:

```
s:=n;
while not (x[s]<n-k+s) do begin
| s:=s-1;
end;
{s — элемент, подлежащий увеличению};
x[s]:=x[s]+1;
for i:=s+1 to n do begin
| x[i]:=x[i-1]+1;
end;
```

2.3.3. Пусть мы решили представлять k-элементные подмножества множества {1...n} убывающими последовательностями длины k, по-прежнему упорядоченными лексико-графически. (Пример : 21 31 32 41 42 43 51 52 53 54.) Как выглядит тогда алгоритм перехода к следующей?

Решение. Ищем наибольшее s, для которого x[s]-x[s+1]>1. (Если такого s нет, полагаем s=0.) Увеличив x[s+1] на 1, кладем остальные минимально возможными (x[t]=k+1-t для t>s).

2.3.4. Решить две предыдущие задачи, заменив лексико-графический порядок на обратный (раньше идут те, которые больше в лексико-графическом порядке).

2.3.5. Перечислить все вложения (функции, переводящие разные элементы в разные) множества {1...k} в {1...n} (предполагается, что k<=n). Порождение очередного элемента должно требовать порядка k действий.

Указание. Эта задача может быть сведена к перечислению подмножеств и перестановок элементов каждого подмножества.

2.4. Разбиения

2.4.1. Перечислить все разбиения целого положительного числа n на целые положительные слагаемые (разбиения, отличающиеся лишь порядком слагаемых, считаются за одно). (Пример: n=4, разбиения 1+1+1+1, 2+1+1, 2+2, 3+1, 4.)

Решение. Договоримся, что в разбиениях слагаемые идут в невозрастающем порядке и сами разбиения мы перечисляем в лексико-графическом порядке.

Разбиение храним в начале массива x[1]...x[n], при этом количество входящих в него чисел обозначим k. Вначале x[1]=...=x[n]=1, k=n, в конце x[1]=n, k=1.

В каком случае x[s] можно увеличить, не меняя предыдущих?

Во-первых, должно быть x[s-1]>x[s] или s=1. Во-вторых, s должно быть не последним элементом (увеличение s надо компенсировать уменьшением следующих). Увеличив s, все следующие элементы надо взять минимально возможными.

```
s:=k-1;
while not ((s=1) or (x[s-1]>x[s])) do begin
| s:=s-1;
end;
{s — подлежащее увеличению слагаемое}
x[s]:=x[s]+1;
sum:=0;
for i:=s+1 to k do begin
| sum:=sum+x[i];
end;
{sum — сумма членов, стоявших после x[s]}
for i:=1 to sum-1 do begin
| x[s+i]:=1;
end;
k:=s+sum-1;
```

2.4.2. По-прежнему представляя разбиения как невозрастающие последовательности, перечислить их в порядке, обратном лексико-графическому (например, для n=4 должно получиться 4, 3+1, 2+2, 2+1+1, 1+1+1+1).

Указание. Уменьшать можно первый член справа, не равный 1; найдя его, уменьшим на 1, а следующие возьмем максимально возможными (равными ему, пока хватает суммы, а последний — сколько останется).

2.4.3. Представляя разбиения как неубывающие последовательности, перечислить их в лексико-графическом порядке. Пример для n=4: 1+1+1+1, 1+1+2, 1+3, 2+2, 4.

Указание. Последний член увеличить нельзя, а предпоследний можно; если после увеличения на 1 предпоследнего члена за счет последнего нарушится возрастание, то из двух членов надо сделать один, если нет, то последний член надо разбить на слагаемые, равные предыдущему, и остаток, не меньший его.

2.4.4. Представляя разбиения как неубывающие последовательности, перечислить их в порядке, обратном лексико-графическому. Пример для n=4: 4, 2+2, 1+3, 1+1+2, 1+1+1+1.

Указание. Чтобы элемент x[s] можно было уменьшить, необходимо, чтобы s=1 или x[s-1]<x[s]. Если x[s] не последний, то этого достаточно. Если он последний, то нужно, чтобы x[s-1]<=(целая часть (x[s]/2)) или s=1.

2.5. Коды Грея и аналогичные задачи

Иногда бывает полезно перечислять объекты в таком порядке, чтобы каждый последующий минимально отличался от предыдущего. Рассмотрим несколько задач такого рода.

2.5.1. Перечислить все последовательности длины n из чисел 1...k в таком порядке, чтобы каждая следующая отличалась от

предыдущей в единственной цифре, причем не более чем на 1.

Решение. Рассмотрим прямоугольную доску шириной n и высотой k. На каждой вертикали будет стоять шашка. Таким образом, положения шашек соответствуют последовательностям из чисел 1...k длины n (s-й член последовательности соответствует высоте шашки на s-й горизонтали). На каждой шашке нарисуем стрелочку, которая может быть направлена вверх или вниз. Вначале все шашки поставим на нижнюю горизонталь стрелочкой вверх. Далее двигаем шашки по такому правилу: найдя самую правую шашку, которую можно подвинуть в направлении нарисованной на ней стрелки, двигаем ее на одну клетку в этом направлении, а все стоящие правее ее шашки (они уперлись в край) разворачиваем кругом.

Ясно, что на каждом шаге только одна шашка сдвигается, т. е. один член последовательности меняется на 1. Докажем индукцией по n, что проходятся все последовательности из чисел 1...k. Случай n=1 очевиден. Пусть n>1. Все ходы поделим на те, где двигается последняя шашка, и те, где двигается не последняя. Во втором случае последняя шашка стоит у стены, и мы ее поворачиваем, так что за каждым ходом второго типа следует k−1 ходов первого типа, за время которых последняя шашка побывает во всех клетках. Если мы теперь забудем о последней шашке, то движения первых n−1, по предположению индукции, пробегают все последовательности длины n−1 по одному разу; движения же последней шашки из каждой последовательности длины n−1 делают k последовательностей длины n.

В программе помимо последовательности x[1]...x[n] будем хранить массив d[1]...d[n] из чисел +1 и −1 (+1 соответствует стрелке вверх, −1 — стрелке вниз).

Начальное состояние: x[1]=...=x[n]=1; d[1]=...=d[n]=1.

Приведем алгоритм перехода к следующей последовательности (одновременно выясняется, возможен ли он, — ответ становится значением булевской переменной p).

```
{если можно, сделать шаг и положить
p:=true, если нет,
положить p:=false}
i:=n;
while (i>1) and
  (((d[i]=1) and (x[i]=n)) or ((d[i]=-1) and
(x[i]=1)))
  do begin
  i:=i-1;
end;
if (d[i]=1 and x[i]=n) or (d[i]=-1 and x[i]=1)
  then begin {i=1}
  p:=false;
end else begin
  p:=true;
  x[i]:=x[i]+d[i];
  for j:=i+1 to n do begin
    d[j]:=-d[j];
  end;
end;
```

Замечание. Для последовательностей нулей и единиц возможно другое решение, использующее двоичную систему. (Именно оно связывается обычно с названием **«коды Грея».**)

Запишем подряд все числа от 0 до 2^n-1 в двоичной системе. Например, для n=3 напишем:

000 001 010 011 100 101 110 111.

Затем каждое из чисел подвергнем преобразованию, заменив каждую цифру, кроме первой, на ее сумму с предыдущей цифрой (по модулю 2). Иными словами, число a[1], a[2], ..., a[n] преобразуем в a[1], a[1]+a[2], a[2]+a[3], ..., a[n−1]+a[n] (сумма по модулю 2). Для n=3 получим:

000 001 011 010 110 111 101 100.

Легко проверить, что описанное преобразование чисел обратимо (и тем самым дает все последовательности по одному разу). Кроме того, двоичные записи соседних чисел отличаются заменой конца 011...1 на конец 100...0, что — после преобразования — приводит к изменению единственной цифры.

Применение кодов Грея

Пусть есть вращающаяся ось и мы хотим поставить датчик угла поворота этой оси. Насадим на ось барабан, выкрасим половину барабана в черный цвет, половину в белый и установим фотоэлемент. На его выходе будет в половине случаев 0, а в половине 1 (т. е. мы измеряем угол «с точностью до 180»).

Развертка барабана:

```
          0       1
->|_|_|_|_|*|*|*|*|<-(склеить бока).
```

Сделав рядом другую дорожку из двух черных и белых частей и поставив второй фотоэлемент, получаем возможность измерить угол с точностью до 90 градусов:

```
 0 0 1 1
 0 1 0 1

|_|_|_|_|*|*|*|*|
|_|_|*|*|_|_|*|*|
```

Сделав третью,

```
 0 0 0 0 1 1 1 1
 0 0 1 1 0 0 1 1
 0 1 0 1 0 1 0 1

|_|_|_|_|*|*|*|*|
|_|_|*|*|_|_|*|*|
|_|*|_|*|_|*|_|*|
```

мы измерим угол с точностью до 45 градусов и т. д. Эта идея имеет, однако, недостаток: в момент пересечения границ сразу несколько фотоэлементов меняют сигнал, и, если эти изменения произойдут не одновременно, на какое-то время показания фотоэлементов будут бессмысленными. Коды Грея позволяют избежать этой опасности. Сделаем так, чтобы на каждом шаге менялось показание лишь одного фотоэлемента (в том числе и на последнем, после целого оборота).

```
0 0 0 0 1 1 1 1
0 0 1 1 1 1 0 0
0 1 1 0 0 1 1 0

 _ _ _ _
|_|_|_|_|*|*|*|*|
|_|_|*|*|*|*|_|_|
|_|*|*|_|_|*|*|_|
```

Написанная нами формула позволяет легко преобразовать данные от фотоэлементов в двоичный код угла поворота.

2.5.2. Напечатать все перестановки чисел 1...n так, чтобы каждая следующая получалась из предыдущей перестановкой (транспозицией) двух соседних чисел. Например, при n=3 допустим такой порядок: 3.2 1 –> 2 3.1 –> 2.1 3 –> 1 2.3 –> 1.3 2 –> 3 1 2 (между переставляемыми числами вставлены точки).

Решение. Наряду с множеством перестановок рассмотрим множество последовательностей y[1]...y[n] целых неотрицательных чисел, у которых y[1]<=0, ..., y[n]<=n–1. В нем столько же элементов, сколько в множестве всех перестановок, и мы сейчас установим между ними взаимно однозначное соответствие. Именно, каждой перестановке поставим в соответствие последовательность y[1]..y[n], где y[i] — количество чисел, меньших i и стоящих левее i в этой перестановке. Взаимная однозначность вытекает из такого замечания. Перестановка чисел 1...n получается из перестановки чисел 1...n–1 добавлением числа n, которое можно вставить на любое из n мест. При этом к сопоставляемой с ней последовательности добавляется еще один член, принимающий значения от 0 до n–1, а предыдущие члены не меняются. При этом оказывается, что изменение на единицу одного из членов последовательности y соответствует перестановке двух соседних чисел, если все следующие числа последовательности y принимают максимально или минимально возможные для них значения. Именно, увеличение y[i] на 1 соответствует перестановке числа i с его правым соседом, а уменьшение — с левым.

Теперь вспомним решение задачи о перечислении всех последовательностей, на каждом шаге которого один член меняется на единицу. Заменив прямоугольную доску доской в форме лестницы (высота i-й вертикали равна i) и двигая шашки по тем же правилам, мы перечислим все последовательности y, причем i-й член будет меняться, лишь если все следующие шашки стоят у края. Надо еще уметь параллельно с изменением y корректировать перестановку. Очевидный способ требует отыскания в ней числа i; это можно облегчить, если помимо самой перестановки хранить функцию i |—> позиция числа i в перестановке (обратное к перестановке отображение) и соответствующим образом ее корректировать. Вот какая получается программа:

```
program test;
  const n=...;
  var
  x: array[1..n] of 1..n; {перестановка}
  inv_x: array[1..n] of 1..n;
  {обратная перестановка}
  y: array[1..n] of integer; {Y[i]<i}
  d: array[1..n] of –1..1; {направления}
  b: boolean;

  procedure print_x;
    var i: integer;
  begin
    for i:=1 to n do begin
      write (x[i], ' ');
    end;
    writeln;
  end;

  procedure set_first;
  {первая перестановка: y[i]=0 при всех i}
    var i : integer;
  begin
    for i:=1 to n do begin
      x[i]:=n+1–i;
      inv_x[i]:=n+1–i;
      y[i]:=0;
      d[i]:=1;
    end;
  end;

  procedure move (var done: boolean);
    var i, j, pos1, pos2, val1, val2, tmp: integer;
  begin
    i:=n;
    while (i>1) and (((d[i]=1) and (y[i]=i–1)) or
        ((y[i]=–1) and (y[i]=0))) do begin
      i:=i–1;
    end;
    done:=(i>1);
    {упрощение связано с тем, что первый
     член нельзя менять}
    if done then begin
      y[i]:=y[i]+d[i];
      for j:=i+1 to n do begin
        d[j]:=–d[j];
      end;
      pos1:=inv_x[i];
      val1:=i;
      pos2:=pos1+d[i];
```

```
    val2:=x[pos2];
    {pos1, pos2 — номера переставляемых
     элементов; val1, val2 — их значения}
    tmp:=x[pos1];
    x[pos1]:=x[pos2];
    x[pos2]:=tmp;
    tmp:=inv_x[val1];
    inv_x[val1]:=inv_x[val2];
    inv_x[val2]:=tmp;
  end;
end;

begin
  set_first;
  print_x;
  b:=true;
  {напечатаны все перестановки
   до текущей включительно;
   если b ложно, то текущая — последняя}
  while b do begin
    move (b);
    if b then print_x;
  end;
end
```

2.6. Несколько замечаний

Посмотрим еще раз на использованные нами приемы. Вначале удавалось решить задачу по такой схеме: определяем порядок на подлежащих перечислению объектах и явно описываем процедуру перехода от данного объекта к следующему (в смысле этого порядка). В задаче о кодах Грея потребовалось хранить помимо текущего объекта и некоторую дополнительную информацию (направления стрелок). Наконец, в задаче о перечислении перестановок (на каждом шаге допустима одна транспозиция) мы применили такой прием: установили взаимно однозначное соответствие между перечисляемым множеством и другим, более просто устроенным. Таких соответствий в комбинаторике известно много. Мы приведем несколько задач, связанных с так называемыми числами Каталана.

2.6.1. Перечислить все последовательности длины 2n, составленные из n единиц и n минус единиц, у которых сумма любого начального отрезка положительна (т. е. число минус единиц в нем не превосходит числа единиц).

Решение. Изображая единицу вектором (1,1), а минус единицу вектором (1,−1), можно сказать, что мы ищем пути из точки (0,0) в точку (n,0), не опускающиеся ниже оси абсцисс.

Будем перечислять последовательности в лексико-графическом порядке, считая, что −1 предшествует 1. Первой последовательностью будет «пила»

1, −1, 1, −1, ...

а последней — «горка»

1, 1, 1, ..., 1, −1, −1, ..., −1.

Как перейти от последовательности к следующей? До некоторого места они должны совпадать, а затем надо заменить −1 на 1. Место замены должно быть расположено как можно правее. Но заменять −1 на 1 можно только в том случае, если справа от нее есть единица (которую можно заменить на −1). Заменив −1 на 1, мы приходим к такой задаче: фиксирован начальный кусок последовательности, надо найти минимальное продолжение. Ее решение: надо приписывать −1, если это не нарушит условия неотрицательности, а иначе приписывать 1. Получаем такую программу:

```
...
type array2n=array[1..2n] of integer;
...
procedure get_next (var a: array2n;
                    var last: Boolean);
  {в a помещается следующая
   последовательность, если она есть
   (при этом last=false), иначе last:=true}
  var k, i, sum: integer;
begin
  k:=2*n;
  {инвариант: в a[k+1..2n] только
   минус единицы}
  while a[k]=-1 do begin k:=k-1; end;
  {k — максимальное среди тех,
   для которых a[k]=1}
  while (k>0) and (a[k]=1) do begin k:=k-1;
end;
  {a[k] — самая правая -1, за которой
   есть 1; если таких нет, то k=0}
  if k=0 then begin
    last:=true;
  end else begin
    last:=false;
    i:=0; sum:=0;
    {sum=a[1]+...+a[i]}
    while i<>k do begin
      i:=i+1; sum:=sum+a[i];
    end;
    {sum=a[1]+...+a[k]}
    a[k]:= 1; sum:=sum+2;
    {вплоть до a[k] все изменено,
     sum=a[1]+...+a[k]}
    while k<>2*n do begin
      k:=k+1;
      if sum>0 then begin
        a[k]:=-1
      end else begin
        a[k]:=1;
      end;
      sum:=sum+a[k];
    end;
    {k=n, sum=a[1]+...a[2n]=0}
  end;
end;
```

2.6.2. Перечислить все расстановки скобок в произведении n сомножителей. Порядок

сомножителей не меняется, скобки полностью определяют порядок действий. (Например, для n=4 есть 5 расстановок ((ab)c)d, (a(bc))d, (ab)(cd), a((bc)d), a(b(cd)).)

Указание. Каждому порядку действий соответствует последовательность команд стекового калькулятора.

2.6.3. На окружности задано 2n точек, пронумерованных от 1 до 2n. Перечислить все способы провести n непересекающихся хорд с вершинами в этих точках.

2.6.4. Перечислить все способы разрезать n-угольник на треугольники, проведя n−2 его диагонали.

Еще один класс задач на перечисление всех элементов заданного множества мы рассмотрим ниже, обсуждая метод поиска с возвратами (backtracking).

2.7. Подсчет количеств

Иногда можно найти количество объектов с тем или иным свойством, не перечисляя их. Классический пример: C(n,k) — число всех k-элементных подмножеств n-элементного множества — можно найти, заполняя таблицу значений функции C по формулам:

C(n,0)=C(n,n)=1 (n>=1)
C(n,k)=C(n−1,k−1)+C(n−1,k) (n>1, 0<k<n);

или по формуле n!/((k!)×(n−k)!). (Первый способ эффективнее, если надо вычислить много значений C(n,k).)

Приведем другие примеры.

2.7.1 (Число разбиений). (Предлагалась на Всесоюзной олимпиаде по программированию 1988 г.) Пусть P(n) — число разбиений целого положительного n на целые положительные слагаемые (без учета порядка, 1+2 и 2+1 — одно и то же разбиение). При n=0 положим P(n)=1 (единственное разбиение не содержит слагаемых). Построить алгоритм вычисления P(n) для заданного n.

Решение. Можно доказать (это нетривиально) такую формулу для P(n):

P(n)=P(n−1)+P(n−2)−P(n−5)−P(n−7)+P(n−12)+P(n−15)+...

(знаки у пар членов чередуются, вычитаемые в одной паре равны (3×q×q−q)/2 и (3×q×q+q)/2).

Однако и без ее использования можно придумать способ вычисления P(n), который существенно эффективнее перебора и подсчета всех разбиений.

Обозначим через R(n,k) (при n>=0, k>=0) число разбиений n на целые положительные слагаемые, не превосходящие k. (При этом R(0,k) считаем равным 1 для всех k>=0.) Очевидно, P(n)=R(n,n). Все разбиения n на слагаемые, не превосходящие k, разобьем на группы в зависимости от максимального слагаемого (обозначим его i). Число R(n,k) равно сумме (по всем i от 1 до k) количеств разбиений со слагаемыми не больше k и максимальным слагаемым, равным i. А разбиения n на слагаемые не более k с первым слагаемым, равным i, по существу, представляют собой разбиения n−i на слагаемые, не превосходящие i (при i<=k). Так что

R(n,k)=сумма по i от 1 до k чисел R(n−i,i) при k<=n;
R(n,k)=R(n,n) при k>=n,

что позволяет заполнять таблицу значений функции R.

2.7.2 (Счастливые билеты). (Задача предлагалась на Всесоюзной олимпиаде по программированию 1989 г.) Последовательность из 2n цифр (каждая цифра от 0 до 9) называется счастливым билетом, если сумма первых n цифр равна сумме последних n цифр. Найти число счастливых последовательностей данной длины.

Решение. (Сообщено одним из участников олимпиады; к сожалению, не могу указать фамилию, так как работы проверялись зашифрованными.) Рассмотрим более общую задачу: найти число последовательностей, где разница между суммой первых n цифр и суммой последних n цифр равна k (k=−9n, ..., 9n). Пусть T(n,k) — число таких последовательностей.

Разобьем множество таких последовательностей на классы в зависимости от разницы между первой и последней цифрами. Если эта разница равна t, то разница между суммами групп из оставшихся n−1 цифр равна k−t. Учитывая, что пар цифр с разностью t бывает 10−|t|, получаем формулу

$$\sum_{t=-q}^{q} (10 - |t|) \times T(n-1, k-t).$$

(Некоторые слагаемые могут отсутствовать, так как k−t может быть слишком велико.)

А. Г. Юдина,
учитель информатики средней школы № 112, Москва

ИНФОРМАТИКА НА ЛОГО ДЛЯ СТАРШЕКЛАССНИКОВ*

8. Рекурсивные функции

Особенно эффективно использование команды OP в сочетании с рекурсией. Разберем несколько примеров из математики, где рекуррентные соотношения и рекурсивные функции играют важную роль.

Факториал натурального числа N — это произведение всех натуральных чисел от единицы до N включительно. Можно предложить два способа вычисления факториала: итеративный и рекурсивный.

Итерация:

N! = 1 × 2 × 3 × ... × N.

Рекурсия:

N! = N × (N − 1)!,
1! = 1.

В первом случае вычисление начинается с единицы, в цикле каждый раз вычисляется очередное натуральное число и накопленное произведение домножается на это число.

```
to факт_ит :N
make "число 1
make "произведение 1
repeat :N
   [make "произведение :произведение * :число
    make "число :число + 1]
op :произведение
end
```

Во втором случае вычисления, наоборот, начинаются с конца и проводятся как бы в обратном направлении. Например:

5! = 5 × 4!,
где 4! = 4 × 3!,
где 3! = 3 × 2!,
где 2! = 2 × 1!,
а 1! = 1 по определению.

Дошли с конца до начала, а теперь — движение в обратную сторону:

2! = 2 × 1 = 2;
3! = 3 × 2 = 6;
4! = 4 × 6 = 24;
5! = 5 × 24 = 120
(5! = 1 × 2 × 3 × 4 × 5 = 120 — по первому способу).

Процесс вычисления вроде бы длиннее и запутаннее. (Кстати, существует много сказок, построенных по такому же принципу — герой обращается с просьбой к персонажу, а тот отсылает его со своей просьбой к другому персонажу, а тот — к следующему... и так до тех пор, пока не найдется кто-то покладистый, и тогда герой сказки проходит по всей цепочке в обратном направлении и достигает желаемого.)

Зато запись рекурсивной функции намного короче и красивее:

```
to факт_рек :N
ifelse :N = 1 [op 1]
   [op :N * факт_рек :N − 1]
end
```

Описание точно следует математической формуле

N! = N × (N − 1)!; 1! = 1.

Можно привести еще ряд примеров, когда используются два варианта записи, соответствующие итерации и рекурсии.

Возведение числа в натуральную степень:

Итерация:

$a^n = a \times a \times ... \times a$ (n раз).

Рекурсия:

$a^n = a \times a^{n-1}$; $a^0 = 1$.

Арифметическая и геометрическая прогрессии:

Итерация:

$a_n = A + d + d + ... + d$ (n раз),
$b_n = B \times q \times q \times ... \times q$ (n раз).

Рекурсия:

$a_n = a_{n-1} + d$; $a_0 = A$.
$b_n = b_{n-1} \times q$; $b_0 = B$.

Сумма ряда:

Итерация:

$S_n = a_1 + a_2 + ... + a_n$.

Рекурсия:

$S_n = S_{n-1} + a_n$; $S_1 = a_1$.

Обратите внимание на то, что в рекурсив-

* Продолжение. Начало см.: Информатика и образование. 1994. № 6.

ном варианте запись всегда содержит условие остановки процесса, выхода из повторений.

Описание рекурсивной функции удобно начинать командой ветвления: если условие остановки процесса выполнено, то считать результатом ..., в противном же случае считать результатом ... (рекурсивный вызов).

Таким образом, в любом случае последней выполняемой командой должна быть OP, возвращающая результат.

Ряд Фибоначчи — это последовательность натуральных чисел:

1, 1, 2, 3, 5, 8, 13, ...

т. е. первый и второй члены ряда — единицы, а каждый из следующих равен сумме двух предыдущих.

$$F_n = F_{n-1} + F_{n-2},$$
$$F_1 = F_2 = 1.$$

Опишем рекурсивную функцию вычисления n-го члена ряда Фибоначчи:

```
to Фиб_рек :n
ifelse or :n = 1 :n = 2
   [op 1]
   [op (Фиб :n - 1) + (Фиб :n - 2)]
end
```

В главе 6 была описана итеративная **функция поиска минимального числа в списке.** Попробуем описать и рекурсивный вариант. Будем рассуждать следующим образом. Если в списке только два элемента, найти минимум достаточно легко. Опишем функцию нахождения минимума из двух чисел:

```
to min_2 :a :b
ifelse :a < :b  [op :a]
                [op :b]
end
```

Если же в списке более двух элементов, то будем считать результатом минимум из двух чисел — первого числа и минимального числа из оставшегося списка (это и есть рекурсивный вызов):

```
to min :cп
ifelse (count :cп) = 2
   [op min_2 first :cп last :cп]
   [op min_2 first :cп min bf :cп]
end
```

И в этом случае рекурсивный вариант короче и красивее.

ЗАДАНИЯ:

а) написать итеративный и рекурсивный варианты для функции возведения числа в натуральную степень (функция от двух аргументов);

б) описать рекурсивную функцию суммирования числового списка и сравнить ее с итеративной процедурой, полученной при решении задания к главе 1;

в) описать итеративный вариант функции вычисления N-го члена ряда Фибоначчи;

г) используя функцию вычисления N-го члена ряда, описать функцию, вычисляющую сумму N первых членов ряда Фибоначчи. Дать три варианта: рекурсивный и итеративный для последовательного накопления суммы; в третьем варианте сначала сформировать список из членов ряда Фибоначчи, а затем использовать функцию суммирования списка.

```
to степ_ит :N :x
make "рез 1
repeat :N [make "рез :рез * :x]
op :рез
end

to степ_рек :N :x
ifelse :N = 0  [op 1]
               [op :x * степ_рек :N - 1 :x]
end

to сумма_рек :cп
ifelse empty? :cп
   [op 0]
   [op (first :cп) + сумма_рек bf :cп]
end


to Фиб_ит :n
make "n1 0 make "n2 1
repeat (:n - 1)  [make "w :n2
                  make "n2 :n2 + :n1
                  make "n1 :w]
op :n2
end


to сумма_Фиб_1 :k
make "sum 0
make "n 1
repeat :k [make "sum :sum + Фиб_ит :n
           make "n :n + 1]
op :sum
end

to сумма_Фиб_2 :n
ifelse :n = 0
   [op 0]
   [op (Фиб_рек :n) + сумма_Фиб_2 :n - 1]
end

to сумма_Фиб_3 :n
make "cп []
make "i 1
repeat :n [make "cп lput (Фиб_рек :i) :cп
           make "i :i + 1]
op сумма_рек :cп
end
```

9. Электронный справочник. Сбор данных

Справочную информацию по какому-либо типу объектов часто представляют в виде таблицы. Например:

Определение солевого раствора с помощью индикатора

Раствор соли	Лакмус	Фенолфталеин	Метиловый оранжевый
Хлорид натрия	Цвет не меняет	Цвет не меняет	Цвет не меняет
Хлорид аммония	Краснеет	Цвет не меняет	Становится розовым
Карбонат натрия	Синеет	Приобретает малиновый цвет	Желтеет

Грибы

Название	Спороносный слой	Съедобность, кулинарная обработка
Масленок	Трубчатый	Засолка, маринование, жарение
Моховик	Трубчатый	Маринование
Волнушка	Пластинчатый	Засолка, жарение
Сморчок	Сумчатый	Отварить и отвар слить, в свежем виде ядовит
Желчный гриб	Трубчатый	Ядовит

Вся информация об объекте называется записью. Каждый столбец (определенные данные, характерные свойства) называется полем.

Таким образом, вся справочная информация может рассматриваться как набор записей, запись — набор полей, количество полей во всех записях одинаково.

Или, в других терминах, все данные — список записей. Все элементы этого списка — списки с одинаковым количеством элементов (полей).

Набор данных:

```
[ [ [поле1] [поле2]...[полеN] ]   запись 1
  [ [поле1] [поле2]...[полеN] ]   запись 2
.....
  [ [поле1] [поле2]...[полеN] ] ] запись K
```

Создадим программу, накапливающую данные в таком виде. Она будет задавать вопросы, соответствующие полям таблицы, и запоминать ответы. Запомнив все данные по одной записи, программа должна быть готова задавать те же вопросы снова.

Входные данные для такой программы — список вопросов и набор уже накопленных данных (список списков ответов).

При первом запуске, когда набора данных еще нет, второй параметр — пустой список ([]).

Результатом работы программы будет набор данных. При следующем запуске программы этот набор станет входным параметром.

Например, пусть мы хотим с помощью нашей программы заполнить электронную телефонную книжку и вопросов всего два: «Как вас зовут?» и «Ваш номер телефона?»

При первом запуске программа сбора данных получает в качестве исходных данных список из двух вопросов и пустой список.

Программа должна задать один за другим пару вопросов, и в результате в наборе данных появится первая запись — список из двух ответов:

```
          поле1              поле2
[ [[Маша Иванова] [999-00-00]] ]
          запись1
```

Затем программа опять задаст те же вопросы, в результате набор данных увеличится еще на одну запись:

```
     поле1              поле2
[ [[Маша Иванова]  [999-00-00]]   запись 1
  [[Котов В.И.]    [777-11-11]] ] запись 2
```

И так далее. Процесс сбора данных можно представить так: многократно повторяется опрос (анкетирование) согласно имеющемуся списку вопросов и полученный список ответов (очередная запись) дописывается в конец списка записей. Закончится работа программы, очевидно, по желанию пользователя, т. е. нужно каждый раз в диалоговом режиме выяснить, продолжать или остановиться.

Так как заранее неизвестно, сколько будет повторений, описание будет рекурсивным.

Список из N записей — это добавление результата анкетирования к списку из (N–1) записей. Записать это можно так:

```
make "набор_данных
  добавление :сп_вопр :набор_данных
```

А для первого раза так:

```
make "набор_данных добавление :сп_вопр []
```

Опишем рекурсивную функцию ДОБАВЛЕНИЕ. Добавление — это приписывание результата анкетирования в конец списка, полученного добавлением.

```
to добавление :сп_вопр :набор_данных
ct
pr [Добавить запись? (д/н)]
ifelse not readchar = "д
  [op :набор_данных]
  [op lput анкета :сп_вопр
    добавление :сп_вопр :набор_данных]
end
```

(Сначала — вопрос о продолжении работы. Если ответ отрицательный, считать результатом имеющийся набор данных. В противном случае считать результатом список, который получится, если приписать набор ответов в хвост того списка, который получится... и т. д.)

Операция LPUT увеличивает набор данных на запись, являющуюся результатом функции АНКЕТА. Вход для нее — список вопросов. Для рекурсивного описания процесс формирования списка ответов можно представить так: список ответов — это ответ на первый вопрос, приписанный в начало списка ответов на все остальные вопросы (который, в свою очередь, является ответом на первый из оставшихся вопросов, приписанным к списку ответов на оставшиеся вопросы... и т. д. до тех пор, пока список вопросов не кончится).

```
to анкета :сп_вопр
ifelse empty? :сп_вопр
  [op []]
  [op fput ответ first :сп_вопр
    анкета bf :сп_вопр]
end
```

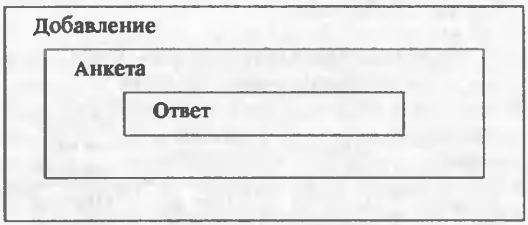
Функцию АНКЕТА можно записать и итеративно (цикл должен повторяться столько раз, сколько в списке вопросов). Если сделаете это самостоятельно, убедитесь в том, что рекурсивная запись выглядит гораздо изящнее.

Описания функций ДОБАВЛЕНИЕ и АНКЕТА похожи, но в первой для формирования списка используется операция LPUT, а во второй — FPUT. Причина этого в том, что в процессе выполнения переменный параметр в первом случае растет (набор данных), а во втором — уменьшается (список вопросов).

Функция АНКЕТА не последний шаг детализации, так как использует результат функции ОТВЕТ. Эта функция должна задавать один вопрос, результатом ее является полученный ответ.

```
to ответ :вопр
pr :вопр
op readlist
end
```

Процесс описания прошел как бы в три этапа, сверху вниз, от общего к деталям. Функция ДОБАВЛЕНИЕ использует функцию АНКЕТА, которая использует функцию ОТВЕТ.



Для работы с электронным справочником необходимо составить список вопросов, в процессе сбора данных накопится список записей. Чтобы труды по вводу текстовой информации не пропали зря, вопросы и ответы нужно сохранить.

LogoWriter предоставляет возможность при необходимости записывать и считывать текстовые и графические файлы, внешние по отношению к файлу LOGOWRITER.COM. Текстовые файлы можно использовать для хранения значений переменных.

Для записи текста в файл его нужно сначала распечатать на экране (например, PR :DATA, если нужно сохранить значение переменной DATA), а затем набрать команду SAVETEXT "имя. Имя может состоять из латинских букв и цифр (не более 8 символов), можно использовать расширение, например SAVETEXT "DATA.TXT.

ЗАДАНИЯ:

а) составить список вопросов для электронного справочника по какой-либо теме (страны, породы собак, информация о ваших знакомых и т. п.);

б) запустить в первый раз процесс сбора данных, накапливая их в некоторой переменной, и ввести несколько записей;

```
(make "набор_данных добавление :сп_вопр [])
```

в) распечатать на экране значение этой переменной и убедиться, что структура полученного списка соответствует задуманной (это должен быть список списков ответов);

г) запустить процесс еще раз, но уже не «с нуля», а задав в качестве второго параметра для функции добавления уже существующий набор данных. Убедиться, что осуществляется пополнение списка записей;

д) сохранить список вопросов и список записей в текстовых файлах.

10. Электронный справочник. Просмотр записей

Предыдущая глава закончилась описанием процесса записи списков с вопросами и ответами в текстовые файлы. Для продолжения работы со справочником необходимо «превратить» эти данные опять в списки.

Каждый текстовый файл нужно распечатать на экране с помощью команды LOADTEXT "имя; затем весь текст выделить, используя команды работы с текстом TOP SELECT BOTTOM. Команда TOP устанавливает курсор в начало текста, а команда BOTTOM — в конец; команда SELECT выделяет текст. Эти действия — программный аналог «ручного» выделения текста при помощи клавиши F1. И наконец, представление данных в виде списка обеспечивают операции PARSE (в переводе — грамматический разбор) и SELECTED (выделенное). Например, MAKE "список PARSE SELECTED.

Если накопленный набор данных достаточно большой, его неудобно просматривать целиком. Нужно организовать просмотр по одной записи, причем ответы предварять соответствующими вопросами.

ЗАДАНИЯ:

а) написать функцию ТЕКСТ-СПИСОК :имя_текста, которая считывает заданный текстовый файл, выделяет распечатанный текст и выдает в качестве результата соответствующий список;

б) написать программу просмотра электронного справочника, работающую следующим образом: на экран выводятся вопросы и соответствующие им ответы по первой записи (для лучшего восприятия можно вопросы и ответы выделять разным цветом — команда SETTC номер цвета); затем — пауза до нажатия любой клавиши (MAKE "R READCHAR). Затем таким же образом выводится информация по второй записи, третьей, и так до конца набора записей. Программа должна быть двухуровневой: процедура просмотра (число повторений равно количеству записей) обращается к процедуре вывода вопросов и ответов по одной записи (число повторений равно количеству вопросов). Способ организации повторений — рекурсия либо итерация — по выбору.

```
to текст-список :имя_текста
ct loadtext :имя_текста
top select bottom
op parse selected
end
```

рекурсия

```
to просмотр :сп_вопр :набор_данных
ct
if empty? :набор_данных [stop]
страница :сп_вопр first :набор_данных
make "r readchar
просмотр :сп_вопр bf :набор_данных
end

to страница :сп_вопр :сп_отв
if empty? :сп_вопр [stop]
settc 12 pr first :сп_вопр
settc 14 pr first :сп_отв
страница bf :сп_вопр bf :сп_отв
end
```

итерация

```
to просмотр! :сп_вопр :набор_данных
repeat count :набор_данных
  [ct
   страница! :сп_вопр first :набор_данных
   make "r readchar
   make "набор_данных bf :набор_данных]
end

to страница! :сп_вопр :сп_отв
repeat count :сп_вопр
  [settc 12 pr first :сп_вопр
   settc 14 pr first :сп_отв
   make "сп_вопр bf :сп_вопр
   make "сп_отв bf :сп_отв]
end
```

11. Электронный справочник. Общая программа

Мы теперь знаем, как организовать сбор данных и просмотр накопленного списка записей. Объединим эти части в общую программу. Для сокращения количества входных данных будем всю необходимую информацию хранить в едином текстовом файле — сначала вопросы, затем данные. Структура общего списка будет следующая:

```
[ [[вопр. 1] [вопр. 2] ... [вопр. N]]
  [[запись 1] [запись 2] ... [запись K]] ]
```

В этом общем списке 2 элемента: список из N вопросов и список из K записей (а каждая запись состоит из N ответов). При самом первом обращении второй список пустой. Исходные данные должны храниться в некотором текстовом файле. После добавления записей этот текстовый файл должен переписываться.

Алгоритм работы программы «Справочник».

1. Задать вопрос об имени текстового файла с данными.

2. Получить ответ.

3. Преобразовать текст в исходные списки.

(П. 1—3 — начальные установки.)

4. Задать вопрос: добавление, просмотр или выход из программы?

5. Получить ответ.

6. В первом случае произвести добавление и перезапись файла с данными, во втором — просмотр.

7. Вернуться к п. 4.

(П. 4—7 — рекурсивный цикл.)

ЗАДАНИЯ:

а) написать процедуру перезаписи файла данных. Старое значение необходимо уничтожить командой ERASEFILE "имя;

б) написать программу «Справочник» и убедиться в том, что она работает с различными исходными данными.

Структура всей программы будет следующая:


```
to запись :имя_перем :имя_текста
ct pr :имя_перем
erasefile :имя_текста
savetext :имя_текста ct
end

to справочник
ct cg ht
pr [В каком файле хранятся данные?]
make "имя first readlist
make "data текст-список :имя ct
make "сп_вопр first :data
make "дан last :data
цикл
end

to цикл
ct settc 15
pr [Добавление – 1; просмотр – 2; выход – 3]
make "r readchar
if :r = "1
   [make "дан добавление :сп_вопр :дан
    запись list :сп_вопр :дан :имя]
if :r = "2 [просмотр :сп_вопр :дан]
if :r = "3 [stop]
цикл
end
```

12. Реализация диалога в виде меню

Один из распространенных способов общения человека с разнообразными программами — выбор из предлагаемого списка вариантов. Например, в предыдущей программе на экран выводится вопрос: к какой части перейти — к добавлению, просмотру данных или окончанию работы? Предполагается, что будет нажата клавиша с соответствующей цифрой. Удобнее и надежнее было бы реализовать диалог в виде меню.

Пусть все варианты будут выведены на экране «в столбик», один под другим. Нужно написать программу, позволяющую «указывать» на тот или иной вариант, передвигая курсор клавишами ↑ и ↓. Программа должна закончиться, когда будет сделан выбор нажатием клавиши ENTER. Эта программа напоминает графический редактор из первой части (занятие 29, см.: ИНФО. 1993. № 5.). Алгоритм следующий:

1) ожидать ввода символа (нажатия на клавишу);

2) анализировать введенный символ: если за клавишей закреплено какое-то действие, выполнить его;

3) вернуться к п. 1.

В графическом редакторе клавиши управляли черепашкой (команды FD, RT и т. п.), в нашей программе нужно управлять курсором — команды CU и CD. Ссылку на клавиши ↑, ↓ и ENTER можно сделать через их ASCII-коды (↑ — код 328; ↓ — код 326; ENTER — код 13). Для таких целей служит операция ASCII. Например:

```
PR ASCII "!
_33
```

Очевидно, для организации повторений нужно использовать рекурсию.

ЗАДАНИЯ:

а) написать процедуру, выводящую заданный список в колонку, один элемент под другим, а затем переводящую курсор на первую строку;

б) написать процедуру, двигающую курсор вверх-вниз со строчки на строчку (пока не будет нажата клавиша ENTER);

в) изменить предыдущую процедуру так, чтобы каждый раз была выделена вся строка (поставить курсор на начало строки — SOL, включить режим выделения — SELECT, отправить курсор в конец строки — EOL; снимает выделение команда UNSELECT).

```
to колонка :cn
make "k count :cn
repeat :k [print first :cn
           make "cn bf :cn]
repeat :k [cu]
end

to вверх-вниз
make "d ascii readchar
if :d = 336 [cd]
if :d = 328 [cu]
if :d = 13 [stop]
вверх-вниз
end

to вверх-вниз!
sol select eol
make "d ascii readchar
unselect
if :d = 336 [cd]
if :d = 328 [cu]
if :d = 13 [stop]
вверх-вниз!
end
```

Рис. 1

Итак, теперь можно все варианты для выбора вывести в колонку и перебирать их, поочередно подсвечивая. Но для того чтобы перейти к тем или иным действиям в зависимости от выбранного варианта, нужно иметь номер этого варианта в списке (т. е. номер строки колонки, на которой была нажата клавиша ENTER). Другими словами, передвигая подсветку по колонке вариантов, необходимо все время отслеживать номер подсвеченной строки. Один из способов такого отслеживания — «закольцевать» перемещения подсветки: если выделена первая строка и нажимается ↑, подсветка должна «перепрыгнуть» на последнюю строку; и, наоборот, с последней строки при нажатии клавиши ↓ подсветка должна перемещаться на первую. Таким образом, изменения нужно внести в ту часть рекурсивной процедуры, где анализируется нажатая клавиша. Первоначальному варианту соответствовала блок-схема центральной части, представленная на рис. 1.

Новый вариант должен соответствовать усложненной блок-схеме (N — номер строки; K — число строк) (рис. 2).

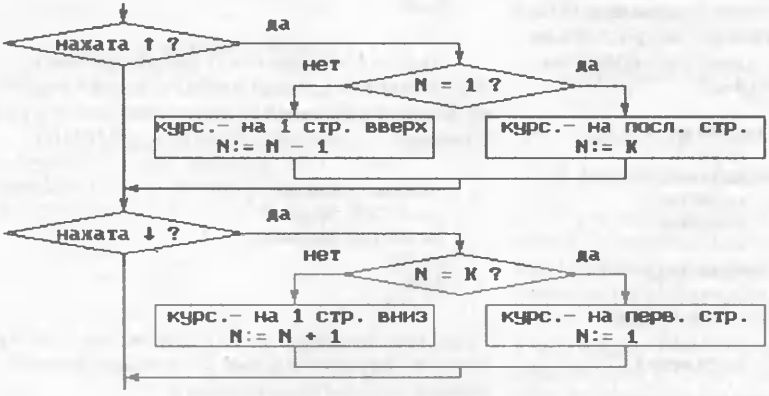
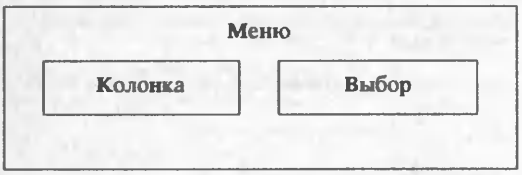


Рис. 2

В окончательном виде процедура МЕНЮ (входные данные — список вариантов) должна распечатывать варианты в колонку, устанавливать начальный номер строки (равный единице, так как уже написанная процедура КОЛОНКА переводит курсор на первую строку) и обращаться к рекурсивной процедуре ВЫБОР. Структура процедуры МЕНЮ:



ЗАДАНИЕ.

Написать и отладить сначала процедуру ВЫБОР (в соответствии с блок-схемой), а затем общую процедуру МЕНЮ.

```
to меню :сп
колонка :сп
make "N 1
выбор
end

to выбор
sol select eol
make "r ascii readchar
unselect
if :r = 336
   [ifelse :N = :k
      [repeat :k - 1 [cu] make "N 1]
      [cd make "N :N + 1]]
if :r = 328
   [ifelse :N = 1
      [repeat :k - 1 [cd] make "N :k]
      [cu make "N :N - 1]]
if :r = 13 [stop]
выбор
end
```

13. Тест. Структура данных

Процедуру МЕНЮ удобно использовать в самых различных ситуациях. Например, при тестировании — когда ответ на вопрос нужно выбрать из списка предлагаемых вариантов. Вот фрагмент теста «Эрудит»:

Компенсировать —
терять
истратить
увеличить
возместить

Аналогия —
случай
явление
свойство
сходство

Адаптироваться —
приспосабливаться
научиться
двигаться
присмотреться

Ироничный —
мягкий
насмешливый
веселый
настоящий

Симптом —
характер
система
желание
признак

Импорт —
собственность
товары
ввоз
фирма

Фрагменты из других тестов:

Буква «т» пропущена в слове:
1. опас.ность
2. мес.ность
3. чудес.ница

Буква «ъ» (твердый знак) пропущена в слове:
1. под.ножка
2. под.езд
3. под.ход

Буква «ь» (мягкий знак) не пишется в слове:
1. нож.
2. ложь.
3. дрожь.

Буква «б» пропущена в слове:
1. кре.кий
2. ро.кий

Округлите число 56,23 до десятков.
56 56,2 50 60

Сколько примерно километров в 2950 м?
2 км 3 км 30 км 29 км

Длина и ширина сада, имеющего форму прямоугольника, равны 600 м и 250 м. Выразите площадь сада в гектарах.
42 га 15 га 1,5 га 150 га

В электронном справочнике мы использовали простую функцию, задающую вопрос и выдающую в качестве результата ответ (см. гл. 9):

```
to ответ :вопр
pr :вопр
op readlist
end
```

Так как теперь ответ не может быть произвольным, а должен выбираться из заданного набора вариантов, функцию ответ нужно изменить, используя процедуру МЕНЮ:

```
to ответ :задание
ct pr first :задание
меню last :задание
op :N
end
```

Здесь задание — это список из двух элементов: первый из них — вопрос, второй — список вариантов ответов:

([[вопрос] [список вариантов]]).

Результат — номер выбранного ответа.

Весь тест — это совокупность тестовых заданий (вопросов и наборов возможных ответов), а также ключ — список номеров правильных ответов. Например, ключ для приведенного фрагмента теста «Эрудит»:

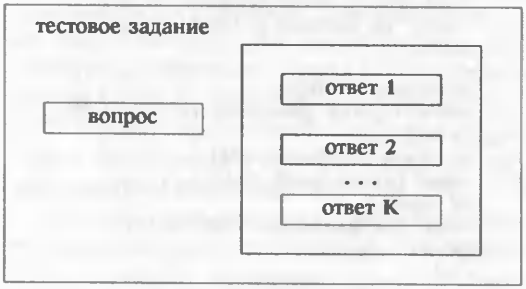
4 4 1 2 4 3

(т. е. правильные ответы — возместить, сходство, приспосабливаться, насмешливый, признак, ввоз).

Таким образом, подобный тест можно представить как список, состоящий из тестовых заданий и ключа (которые сами — тоже списки).

Поставим себе целью написать тестирующую программу, которая будет предлагать вопросы и наборы возможных ответов к ним, а в конце сообщать результаты тестирования. Исходными данными для такой программы будут, конечно, различные тесты.

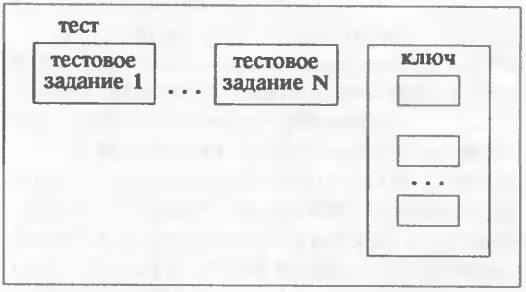
Структура данных для тестирующей программы



Тестовое задание — список из двух элементов:

1) вопрос;

2) список возможных ответов.



Тест — список из тестовых заданий и ключа. Ключ, в свою очередь, — список номеров правильных ответов (элементов в нем столько, сколько тестовых заданий).

ЗАДАНИЯ:

а) проверить функцию ОТВЕТ и нарисовать ее структуру;

б) подготовить текстовый файл с каким-либо тестом. Тест должен быть структурирован с помощью квадратных скобок: каждое тестовое задание, а также ключ должны быть заключены в квадратные скобки; вопрос (если он не состоит из одного слова) и список вариантов ответов в каждом тестовом задании должны быть в квадратных скобках; варианты ответов, не состоящие из одного слова, также должны быть заключены в квадратные скобки;

в) используя функцию, преобразующую текст в список (см. гл. 10), и операцию COUNT, убедиться, что подготовленный список имеет необходимую структуру. Например, команда PR COUNT текст-список "TEST.TXT должна выдать число, на единицу большее, чем выдаст команда PR COUNT LAST текст-список "TEST.TXT; а команда PR COUNT ITEM ? текст-список "TEST.TXT должна выдавать двойку, если вместо ? подставлять номера тестовых заданий.

14. Тест. Структура программы

Программа ТЕСТ должна выполнить три действия:

1) преобразовать заданный текстовый файл в список;

2) выдать одно за другим все тестовые задания и накопить список номеров указанных ответов;

3) сравнить этот список с ключом и выдать результат тестирования.

```
to тест :имя_текста
make "T текст-список :имя_текста
вопросы bl :T
результаты last :T
end
```

(T — это список

[[задание] ... [задание] [ключ]]).

Процедура ВОПРОСЫ — это цикл, в котором обрабатываются поочередно все тестовые задания (с помощью функции ОТВЕТ), а результаты накапливаются в список.

Процедура РЕЗУЛЬТАТЫ должна выдать результат поэлементного сравнения двух списков: ключа и списка номеров, сформированного предыдущей процедурой. Функция сравнения списков уже есть (см. задание к гл. 5).

Структура программы ТЕСТ

(на предыдущих схемах — структура данных)

Процедура МЕНЮ и функции ТЕКСТ-СПИСОК и СРАВНЕНИЕ использованы в программе ТЕСТ, как уже готовые строительные блоки. Их можно скопировать на лист с новой программой со старых листов, но есть и другой путь. Часто используемые средства — процедуры или функции — можно держать на отдельных листах и делать их доступными с любого другого листа с помощью команды GETTOOLS "имя (нужно указать имя листа, где находятся нужные инструменты; tools — инструменты).

Если подготовить несколько тестов, можно использовать процедуру МЕНЮ и для выбора теста. Например, пусть в файлах G.TST, H.TST и M.TST хранятся тесты по географии, истории и математике. Тогда для выбора теста можно вставить фрагмент:

```
. . .
pr [Выбор теста:]
меню [география история математика]
if :N = 1 [make "имя_текста "g.tst]
if :N = 2 [make "имя_текста "h.tst]
if :N = 3 [make "имя_текста "m.tst]
. . .
```

ЗАДАНИЯ:

а) написать и отладить процедуру ВОПРОСЫ (исходные данные — список тестовых заданий), которая вырабатывает список номеров ответов, приписывая каждый раз очередной результат функции ОТВЕТ в конец уже накопленного списка (LPUT). Сначала этот список пустой;

б) написать процедуру РЕЗУЛЬТАТЫ, выдающую оценку (или количество правильных и неправильных ответов);

в) собрать и проверить общую процедуру ТЕСТ. Оформить программу, используя смену цвета текста (SETTC), отступы (TAB) и т. д.

```
to вопросы :сп_заданий
make "сп_отв []
make "K count :сп_заданий
repeat :K
[make "сп_отв
 lput ответ first :сп_заданий :сп_отв
 make "сп_заданий bf :сп_заданий]
end

to результаты :ключ
make "оценка сравнение :ключ :сп_отв
ct pr[]
pr [Твой результат:] pr[]
insert [правильных ответов ____]
pr :оценка
insert [неправильных ответов __]
pr :K – :оценка
end
```

(Продолжение следует)

А. Н. Маслов,
кандидат физико-математических наук,
директор негосударственного образовательного учреждения «Луч»

ДЛЯ СОВРЕМЕННОЙ ШКОЛЫ ЕСТЬ ВЫБОР

«Работа, проведенная Черчем и другими, относит отождествление эффективной вычислимости с рекурсивностью далеко за пределы рабочей гипотезы. Но попытка маскировать это отождествление определением скрывает тот факт, что сделано фундаментальное открытие, касающееся математизационных возможностей Homo Sapiens, и делает нас слепыми в отношении необходимости постоянно подтверждать это отождествление».

Эмиль Л. Пост

Вопрос о том, какие знания нужны современному человеку, является скорее не педагогическим, а политическим. Не случайно Департамент труда США сформулировал пять базовых требований. Это:

1) уметь организовать доступные ресурсы для обеспечения наибольшей эффективности производства;

2) плодотворно сотрудничать с другими участниками производственного процесса;

3) собирать, оценивать и использовать информацию;

4) понимать и уметь использовать взаимосвязи между компонентами сложных производственных систем;

5) уметь осваивать и использовать постоянно расширяющийся спектр производственных технологий.

Указанные требования касаются навыков и умений, позволяющих молодому человеку жить и работать в условиях развитого индустриального общества.

На мой взгляд, в этих требованиях нет и тени личностного аспекта. Но ведь школа должна готовить не просто будущего работника, а личность. И уже как следствие, будут удовлетворены высокие требования работодателей.

Итак, сегодня принципиальные задачи обучения в школе можно сформулировать следующим образом:

- развитие памяти;
- развитие логики;
- воспитание отношения к труду как к процессу коллективного творчества.

Решить эти задачи возможно в процессе интеллектуальной коллективной работы учеников и учителей, в том числе и на занятиях информатикой.

В современных школьных курсах информатики и математики используется символика математической логики. Но объяснению символики и логических понятий в существующих учебниках информатики не уделяется достаточно внимания. По моему мнению, обучение школьников основам математической логики, а также изучение понятия «алгоритм», невозможно без развития «живого логического мышления... которое принципиально отличается от работы любого вида вычисляющих автоматов».

Российское просвещение имеет двухсотлетний опыт преподавания логики в гимназиях и училищах. Современные элитные школы Москвы уже ввели курс логики. В России также необходимо возобновить обучение занимательной и классической логике в младших классах как составной части курса информатики. Хотя цели преподавания в элитных и обычных школах

могут различаться, но методы обучения вполне могут совпадать.

Учебная литература небольшими тиражами издается. Книги по логике и информатике можно заказать в КЦО «Луч» и получить по почте. Задачник по логике готовится к печати.

Программа обучения логике должна формироваться с учетом дошкольной подготовки детей, а задачи — составляться с учетом местной культуры и обычаев, не повторяя механически споры древних философов.

Тогда компьютеризация и информатизация образования станут не целью, а средством реальных преобразований в школе.

Внедрение компьютерных технологий в сферу образования можно уже назвать историческим процессом. С начала программы информатизации образования (середина 80-х гг.) в школу поставляли, как правило, компьютеры отечественного производства и «Ямахи». Оснащение школ IBM-совместимой компьютерной техникой началось с осуществления «Пилотного проекта», результатом которого была государственная поставка в 1990—1991 гг. около 1000 комплектов моделей из минимальной конфигурации персональных компьютеров серии PS-2 фирмы IBM.

В настоящее время ситуация изменилась. Централизованные закупки и распределение компьютеров практически повсеместно, за исключением отдельных регионов, почти полностью прекращены.

Вопросы финансирования поставок оборудования в школы, техникумы, гимназии, лицеи и др. решаются либо на региональном уровне, либо на уровне конкретного учебного заведения.

Что же предлагает современный компьютерный рынок школе?

Прежде всего, помимо собственно компьютеров, системы мультимедиа, сетевые технологии и компьютерные телекоммуникации.

Появление у персональных компьютеров принципиально новых качеств и возможностей интегрировано в понятие «мультимедиа», которое подразумевает взаимодействие визуальных и звуковых эффектов под управлением интерактивного программного обеспечения. Технические средства для обеспечения этих эффектов:

- дисководы для компакт-дисков;
- компакт-диски (CD-ROM);
- аудиоадаптеры (а также микрофоны, динамики, наушники);
- аппаратные и программные средства для ввода, вывода и обработки фрагментов видеофильмов.

Естественно, что работа с этими средствами требует соответствующего программного обеспечения.

Существует мнение западных исследователей потребительского рынка, что наибольшее число мультимедиа-продуктов представляет интерес и будет разрабатываться, прежде всего, для сферы образования, а уже потом для рекламы и моделирования производственных процессов.

Необходимость информационного обмена в современном обществе вызывает развитие и внедрение компьютерных телекоммуникаций в различные сферы деятельности, прежде всего в бизнес, банковское дело, средства массовой информации, науку и непосредственно в процесс обучения. Телекоммуникационные средства передачи данных с одного компьютера на другой при помощи модема и телефонной сети формируют новую методологию обучения.

Для использования телекоммуникационной сети в школе нужны не только IBM-совместимый компьютер, модем, программное и методическое обеспечение, но и профессионально обученный инициативный преподаватель.

Применение сетевых технологий в школе уже стало реальностью, и спектр используемых сетевых плат и пакетов достаточно широк.

Школе есть что выбирать. При этом необходимо учитывать, что вся техника, в том числе и IBM-совместимая, которую покупает школа, производится и продается самыми разными фирмами и организациями. К сожалению, такие категории, как авторитет фирмы и ее репутация, еще только начинают входить в наш обиход.

Сейчас начались поставки в школы компьютеров Macintosh. Хотя технические преимущества этих компьютеров неоспоримы, массовое использование IBM-совме-

стимых компьютеров во всех сферах деятельности обеспечивает школьнику, использующему эту технику, более широкие возможности для получения работы. Однако развитие компьютерной техники происходит настолько бурно, что появление принципиально новых моделей компьютеров значительно изменит содержание предмета ОИВТ в школе.

Итак, для школы появилась возможность выбора, но вместе с ней и новые проблемы:

- разнотипность техники в школах;
- отсутствие системы обмена опытом;
- высокая стоимость программно-методических комплексов при низком качестве технической и содержательной проработки;
- нехватка книг, учебников и методических пособий;
- отсутствие запасных частей для ремонта и обслуживания техники;
- текучесть кадров учителей информатики;
- необходимость регулярной переподготовки учителей на новых технических средствах и программных продуктах.

При существующей ситуации в стране эти проблемы не находят централизованного решения. Обслуживание и использование дорогостоящей вычислительной техники во всех сферах деятельности и в образовании прежде всего требуют специальных знаний и подготовки.

Поэтому вопрос, чему обучать и школьников и учителей, надо решать всем заинтересованным организациям сообща.

Существенную помощь в решении этих проблем могут оказать негосударственные образовательные учреждения. Одним из таких учреждений является компьютерный центр обучения «Луч», который, в соответствии с уставом и лицензией Министерства образования РФ, занимается повышением квалификации преподавателей информатики и других пользователей компьютеров.

КЦО «Луч» проводит обучение учителей информатики на двухнедельных курсах (80 часов) с использованием современных программных и технических средств по следующей программе (приведена в сокращенном виде).

Темы занятий:

1. Операционная система MS DOS, утилиты и архиваторы.

2. Графическая оболочка MS Windows.

3. Текстовый редактор MS Word for Windows.

4. Электронные таблицы MS Excel for Windows.

5. Система подготовки презентаций PowerPoint (с использованием аудио- и видеоэффектов).

6. Графический редактор Story Board Plus.

7. Компьютерные телекоммуникации (возможности модемов, пакеты QL2FAX, MTEZ, электронная почта, телеконференции, электронные доски объявлений, удаленный доступ и удаленная обработка баз данных).

8. Локальные сети. Введение в компьютерные сети и навыки практической работы (Novell, Windows for WorkGroups).

9. Программное обеспечение компьютеров. Выбор технических и программных средств для кабинета информатики. Санитарно-гигиенические правила при работе с компьютером.

10. Демонстрация обучающих программ по общеобразовательным предметам, программ развивающего характера.

11. Основы логики и теории алгоритмов в курсе информатики.

12. Средства MultiMedia (демонстрация звуковых и видеоплат, программ обучения на компакт-дисках, включая звуковые и музыкальные фрагменты).

Программа обучения может быть изменена в зависимости от уровня подготовки и пожеланий заказчика.

Слушатели помимо обучения могут получить консультации и по вопросам, связанным с закупкой нового оборудования и запчастей, обеспечения учебных заведений книгами и учебно-методическими пособиями, новыми программными средствами и расходными материалами. В решении этих проблем помогает КЦО «Луч».

Помимо организации дополнительного образования КЦО «Луч» занимается издательской деятельностью, распространением духовных и культурных ценностей, книг и иных носителей информации.

Наш адрес: 125438, Москва, а/я 51.

Телефон: (095) 454-00-35.

Програ

Обеспечение системными, инструментальными, программно-педагогическими и административными средствами для IBM-совместимой и техники типа УКНЦ, УКНЦ-01, КУВТ-86, ДВК. Каталог программ высылается бесплатно

Учебные пособия

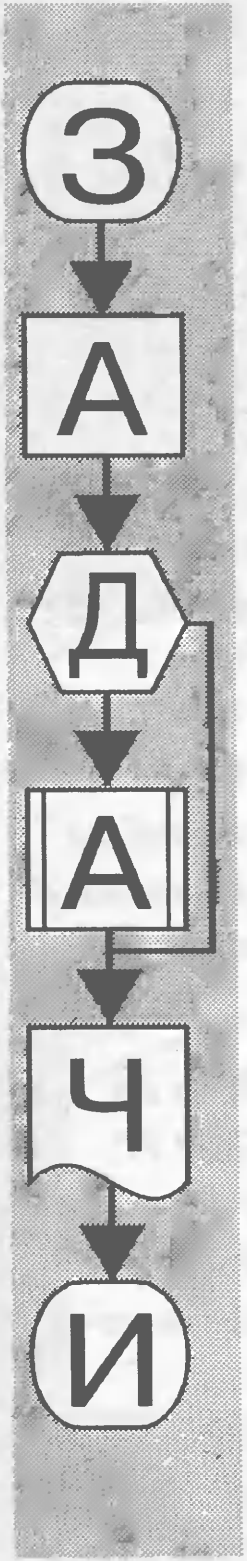
Карасева Т.В. Сборник задач и упражнений по основам информатики и вычислительной техники.

Поддубная Л.М. и др. 10 уроков на IBM PC.

Каймин В.А. Основы компьютерной технологии.

ЭКСИ

АОЗТ Фирма "ЭКСИ"



В. М. Нечаев,
методист лаборатории информатики
Московского института повышения квалификации работников образования

ОДНА ЗАДАЧА — ТРИ ЯЗЫКА

С внедрением в школу развитой вычислительной техники перед учителем встает много проблем, одна из которых — переход от традиционного БЕЙСИКА к возможностям более современных языков. Причем этот переход не сводится просто к изучению новых операторов, а представляет собой прежде всего изменение самого стиля программирования, разработку *структурированных программ*.

Вниманию читателей журнала предлагается первая из серии статей, объединенных общим названием и предназначенных для демонстрации этого стиля программирования. В каждой статье рассматривается одна школьная задача на программирование. Это рассмотрение включает:

- постановку задачи;
- описание алгоритма ее решения;
- тексты программы на трех языках — БЕЙСИКЕ, QBasic, TurboPascal;
- подробный построчный комментарий текстов программ на QBasic и TurboPascal;
- набор заданий для самостоятельной работы учащихся.

Задача «Треугольник»

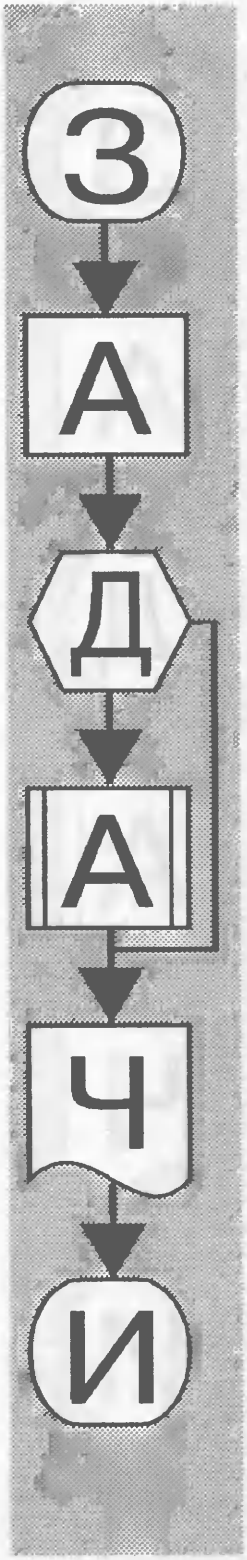
Имеется некоторый плоский треугольник, длины сторон которого известны. Определить, является ли этот треугольник прямоугольным.

Простая программа на БЕЙСИКЕ

```
 10 CLS
 20 INPUT "Сторона a ==>"; r(1)
 30 INPUT "Сторона b ==>"; r(2)
 40 INPUT "Сторона c ==>"; r(3)
 50 r(4)=r(1) : r(5)=r(2)
 60 FOR i=1 TO 3
 70 IF r(i)>=r(i+1)+r(i+2) THEN PRINT "Это не треугольник!": GOTO 20
 80 NEXT i
 90 FOR i=1 TO 3
100 IF r(i)^2=r(i+1)^2+r(i+2)^2 THEN 130
110 NEXT i
120 PRINT "В треугольнике нет прямого угла!": END
130 PRINT "Это прямоугольный треугольник!"
140 PRINT "Его гипотенуза — сторона "; r(i)
```

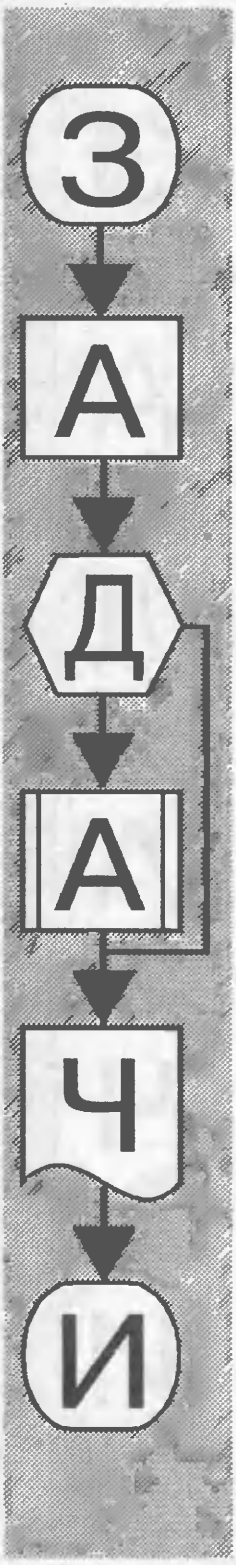
Тоже БЕЙСИК, но структурно

```
10 REM Программа triangle .....
20 REM Исследует треугольник на прямоугольность
30 REM Переменные: r(i) — массив сторон треугольника
40 REM f — т. н. флаг для сигнала об успехе поиска

100 CLS : REM Основная программа .....
110 GOSUB 200 : REM Вводим стороны треугольника (с проверкой)
120 GOSUB 300 : REM Ищем прямой угол в треугольнике
130 GOSUB 400 : REM Выводим результат поиска
190 END : REM Конец программы .....

200 REM Подпрограмма ввода сторон (с проверкой)
210 INPUT "Сторона a ==>"; r(1)
220 INPUT "Сторона b ==>"; r(2)
230 INPUT "Сторона c ==>"; r(3)
240 r(4)=r(1) : r(5)=r(2)
250 FOR i=1 TO 3
260 IF r(i)>=r(i+1)+r(i+2) THEN PRINT "Это не треугольник!": GOTO 210
270 NEXT i
290 RETURN : REM Конец подпрограммы .....

300 REM Подпрограмма поиска прямого угла.
310 f=0 : REM Флаг поиска поначалу опущен.
320 FOR i=1 TO 3
330 IF r(i)^2=r(i+1)^2+r(i+2)^2 THEN f=1 : GOTO 390 : REM Нашли!
330 NEXT i
390 RETURN : REM Конец подпрограммы .....

400 REM Подпрограмма вывода результата
410 IF f=0 THEN PRINT "В треугольнике нет прямого угла!": GOTO 490
420 PRINT "Это прямоугольный треугольник!"
430 PRINT "Его гипотенуза — сторона "; r(i)
490 RETURN : REM Конец подпрограммы .....
```



Этот пример показывает, что и на старом БЕЙСИКЕ можно писать структурированные программы, хотя бы и в элементарной форме. Услышав где-нибудь высказывание о том, что «язык БЕЙСИК неструктурированный», можно, пожалуй, возразить. Структурность характеризует не сам язык, а стиль программирования на нем. Вышеприведенное высказывание, очевидно, надо понимать не буквально, а в том смысле, что Бейсик не предоставляет широкого спектра возможностей для развития такого стиля.

В этом отношении гораздо большим потенциалом обладает QBasic, но и здесь в конечном счете все определяется профессиональным уровнем того, кто пишет программу. Может ли он разбить всю задачу на самостоятельные фрагменты и правильно организовать последовательность их исполнения и взаимодействия, в чем и состоит сущность структуры? Пожалуй, этот вопрос уже выходит за рамки чистого программирования и связан с общекультурными навыками умственного труда.

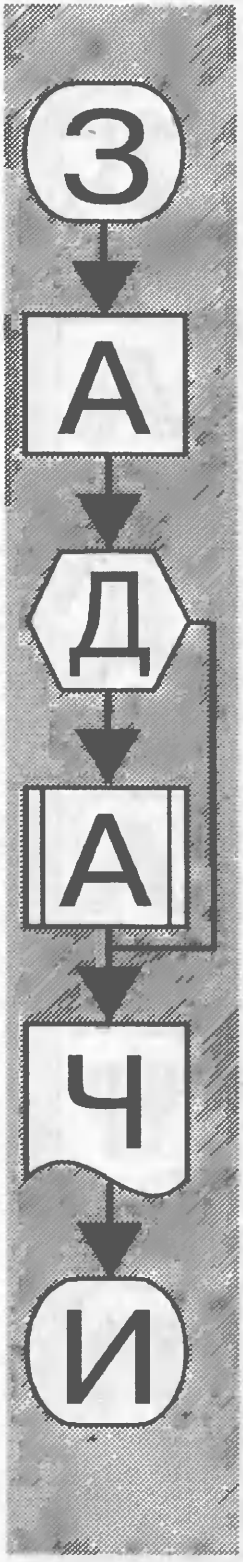


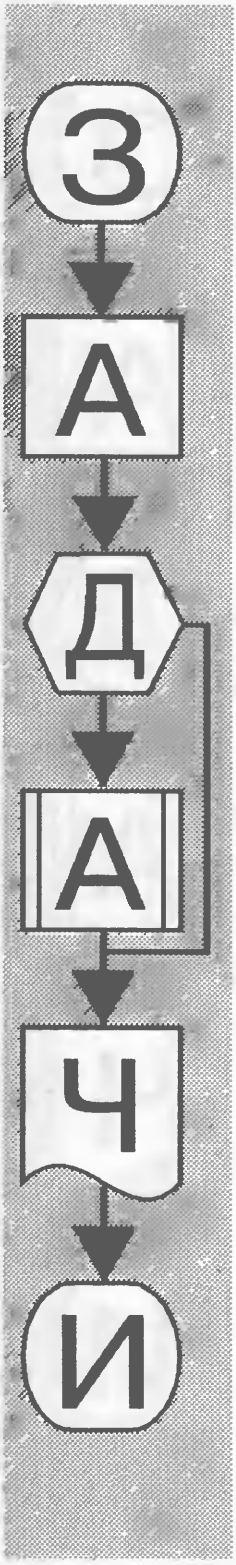
Программа на QBasic

```
' Программа triangle. Исследует треугольник на прямоугольность

' Объявляем имена процедур .....
DECLARE SUB init( )      ' Процедура ввода сторон треугольника
DECLARE SUB process( )   ' Процедура поиска прямого угла
DECLARE SUB result( )    ' Процедура вывода результата

' Объявляем имена переменных .....
DIM SHARED r(1 TO 5)  ' r(i) — массив сторон треугольника
DIM SHARED i,f        ' i — номер очередной стороны
                      ' f — флаг для сигнала об успехе поиска

' Начинаем программные действия .....
CLS
CALL init       ' Вводим стороны треугольника (с проверкой)
CALL process    ' Ищем прямой угол
CALL result     ' Выводим результат поиска
END             ' Конец программы .....

' Описываем работу объявленных процедур
SUB init            ' Процедура ввода сторон треугольника
ret: INPUT "Сторона a ==>"; r (1)
INPUT "Сторона b ==>"; r(2)
INPUT "Сторона c ==>"; r(3)
r(4)=r(1): r(5)=r(2)
FOR i=1 TO 3
  IF r(i) >= r(i+1)+r(i+2) THEN
    PRINT "Это не треугольник!"
    GOTO ret    ' Повторить ввод
  END IF        ' Конец условного оператора
NEXT i
END SUB         ' Конец процедуры .....

SUB process         ' Процедура поиска прямого угла
f = 0               ' Флаг поиска поначалу опущен
FOR i=1 TO 3
  IF r(i)*r(i)=r(i+1)*r(i+1)+r(i+2)*r(i+2) THEN
    f=1             ' Нашли! Поднять флаг!
    EXIT FOR        ' Больше искать не надо
  END IF            ' Конец условного оператора
NEXT i
END SUB             ' Конец процедуры .....

SUB result          ' Процедура вывода ответа
IF f=0 THEN PRINT "В треугольнике нет прямого угла!": EXIT SUB
PRINT "Это прямоугольный треугольник!"
PRINT "Его гипотенуза — сторона "; r(i)
END SUB             ' Конец процедуры .....
```

Программа на TurboPascal

```
PROGRAM triangle;{ Исследует треугольник на прямоугольность }
    USES CRT;
    { Объявляем имена переменных ..........................................}
    VAR r: ARRAY[1..5] OF REAL;
      i: BYTE;       { Номер очередной стороны }
      f: BOOLEAN;    { Флаг для сигнала об успехе поиска }
    { Объявляем процедуры и описываем их работу ..........................}
    PROCEDURE init;  { Процедура ввода сторон (с проверкой) }
    LABEL ret;       { Метка внутри процедуры для повторного ввода }
    BEGIN
ret: WRITE ('Сторона a ==>'); READLN r[1];
    WRITE ('Сторона b ==>'); READLN r[2];
    WRITE ('Сторона c ==>'); READLN r[3];
    r[4]:=r[1]; r[5]:=r[2];
    FOR i:=1 TO 3 DO
      IF r[i] >= r[i+1]+r[i+2] THEN
        BEGIN
        WRITELN "Это не треугольник!";
        GOTO ret;    { Повторить ввод }
        END;       { Конец условного оператора }
    END;           { Конец процедуры.....................................}
    PROCEDURE process; { Процедура поиска прямого угла }
    BEGIN
      f:=FALSE;    { Флаг поначалу опущен }
      FOR i:=1 TO 3 DO
        IF SQR(r[i])=SQR(r[i+1])+SQR(r[i+2]) THEN
          BEGIN
          f:=TRUE; { Нашли! Поднять флаг! }
          EXIT; { Больше искать не надо }
          END; { Конец условного оператора }
    END;           { Конец процедуры.....................................}
    PROCEDURE result;   { Процедура вывода результата }
    BEGIN
      IF NOT(f) THEN
        BEGIN
        WRITELN ('В треугольнике нет прямого угла!');
        EXIT;      { Больше искать не надо }
        END;       { Конец условного оператора }
      WRITELN ('Это прямоугольный треугольник!');
      WRITELN ('Его гипотенуза — сторона ', r[i]:3:1);
    END;           { Конец процедуры.....................................}
    { Начинаем программные действия }
    BEGIN
    ClrScr;
    init;
    process;
    result;
    END.           {Конец программы}
```





Алгоритм решения задачи

Задача очень простая. Сначала вводятся исходные данные — длины сторон. Затем проверяется выполнение, так сказать, обратной теоремы Пифагора: если сумма квадратов двух сторон треугольника равна квадрату третьей стороны, то этот треугольник прямоугольный, причем гипотенузой является третья сторона.

Программные действия естественным образом разбиваются на три обособленные части: ввод данных, поиск прямого угла и вывод результата этого поиска. Каждую из частей удобно оформить в виде отдельной подпрограммы.

Таким образом, начинает выявляться структура программы. Центральное ее место занимает основной блок, где перечисляются в нужном порядке только лишь имена (или адреса, если речь идет о старом БЕЙСИКЕ) подпрограмм. Описания же самих действий, составляющих тело каждой подпрограммы, выносятся на периферию — например, после текста основного блока, как в QBasic, или наоборот, перед этим текстом, как в TurboPascal.

```
ПРОГРАММА треугольник
(Основной блок):
ВВОД СТОРОН (Вызываем на исполнение подпрограмму ввода)
ПОИСК ПРЯМОГО УГЛА    (Вызываем на исполнение
                       подпрограмму поиска)
ВЫВОД РЕЗУЛЬТАТА      (Вызываем на исполнение
                       подпрограмму вывода)

(Описание подпрограмм)
ПОДПРОГРАММА ВВОДА
   (Действия, реализующие ввод)
ПОДПРОГРАММА ПОИСКА
   (Действия, реализующие поиск)
ПОДПРОГРАММА ВЫВОДА
   (Действия, реализующие вывод)
```

Когда программа структурирована, ее легко читать и модифицировать, подстраивая под изменяющиеся условия задачи. Дело в том, что в таком виде становятся обозримыми шаги, приводящие к конечной цели. Они представляют собой обособленные этапы решения. Конкретный же механизм их работы приводится отдельно, чтобы не перегружать текст основной программы техническими деталями. При этом остается возможность заменять этот механизм на другой, не затрагивая всю программу.

Первая подпрограмма, в которой осуществляется ввод сторон треугольника, представлена в приведенном примере в виде ПРОЦЕДУРЫ с именем **init**. Помимо собственно ввода в ней содержится еще проверка исходных данных, поскольку не любые три стороны могут образовывать треугольник, а именно ни одна сторона не должна быть больше, чем две другие вместе взятые. Если такое случилось, то ввод должен быть повторен.

Вторая подпрограмма, осуществляющая поиск прямого угла в треугольнике, представляет собой еще одну ПРОЦЕДУРУ, но уже с другим именем — **process**. В ней все стороны, введенные с помощью процедуры **init**, одна за одной проверяются на выполнение обратной теоремы Пифагора. Как только обнаружится, что

очередная проверка дала положительный результат, так это событие фиксируется и известие о нем передается в основную программу. Механизм этой фиксации состоит в так называемом поднятии флага, т. е. в присваивании некоторой переменной определенного значения.

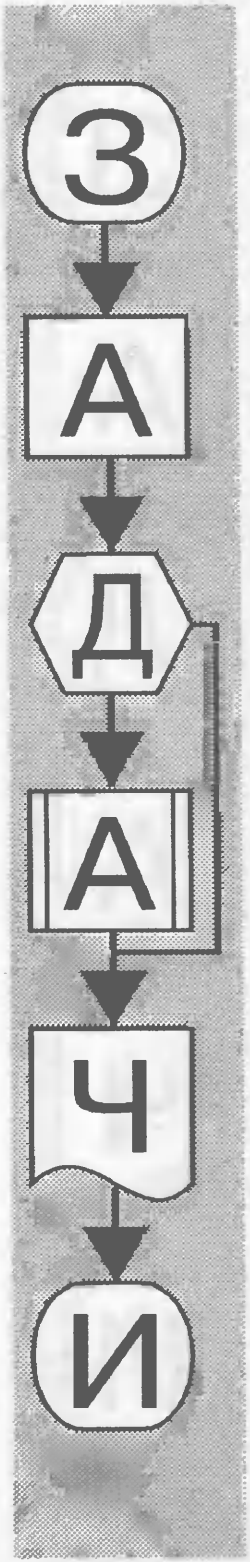
В третьей подпрограмме, представленной ПРОЦЕДУРОЙ с именем **result**, происходит анализ того значения, которое имеет флаговая переменная, переданная процедуре при ее вызове из основной программы. В соответствии с результатом этого анализа производится вывод ответа на поставленную задачу.

Как это обычно и бывает, приведенное программное решение не является единственно возможным. В частности, вторую подпрограмму можно было бы оформить в виде ФУНКЦИИ, например с именем **flag**, которая принимала бы одно из двух возможных значений, скажем 1 и 0 или истина (true) и ложь (false).

Далее, саму работу этой подпрограммы, независимо от того является она процедурой или функцией, можно было бы построить по другому принципу. Вместо того чтобы в каждой стороне треугольника искать гипотенузу, следовало бы сначала найти наибольшую из сторон и только для нее проверить теорему Пифагора.

Да и количество подпрограмм могло бы быть иным, большим или меньшим. Скажем, проверку исходных данных можно было бы выделить в отдельную процедуру или функцию.

Однако во всех этих вариантах структурированный вид программы имеет преимущество большей ясности перед простым, так сказать, «потоком сознания» программиста. В рассмотренном примере это преимущество не имеет большой ценности, так как сама задача предельно проста. Более того, использование подпрограмм представляется даже излишним — достаточно взглянуть на размер структурированной и простой программы. Но для больших по объему и сложных по логике задач этому подходу нет альтернативы. Приведенная же задача имела целью в основном дать возможность для демонстрации такого подхода.



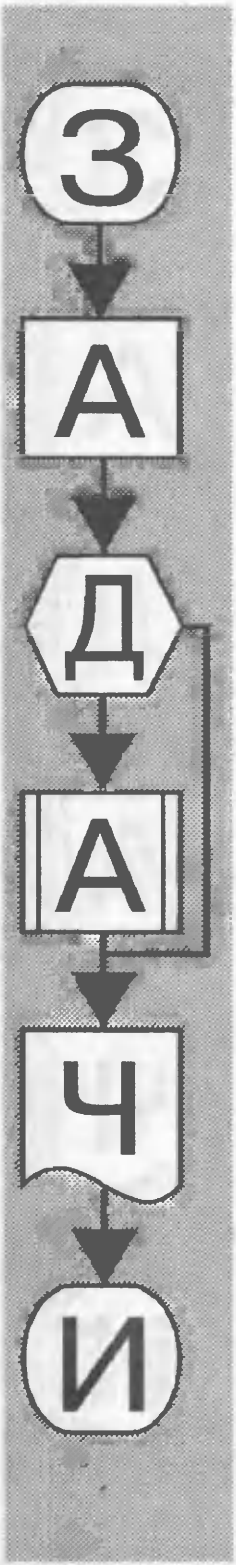
Комментарии к тексту программы на QBasic

```
DECLARE SUB init ( )
DECLARE SUB process ( )
DECLARE SUB result ( )
```

В этих трех строчках мы объявляем (DECLARE) имена тех процедур (SUB), которые будут использоваться в программе. Пустые скобки после имени указывают на то обстоятельство, что в эти процедуры при их вызове из основной программы не будут передаваться никакие параметры.

```
DIM SHARED r(1 TO 5)
DIM SHARED i, f
```

В этих двух строчках мы объявляем имена переменных (DIM), используемых в программе. Ими будут две простые числовые величины i и f, а также одномерный массив r из пяти элементов (1 TO 5). Служебное слово SHARED дополнительно придает этим переменным статус «разделяемых». Это означает, что к ним раз-



решен доступ со стороны всех процедур и функций программы. Например, если переменная f фигурирует в процедуре result, то ее значение внутри этой процедуры будет тем же самым, что и в основной программе или в процедуре process. Если бы переменная f не имела статуса разделяемой, то ее значения внутри каждой процедуры и в основной программе были бы свои собственные, никак не связанные друг с другом.

На этом декларативный раздел заканчивается, и мы приступаем к самим программным действиям основного блока.

```
CLS
```

Это имя стандартной процедуры очистки экрана. Она встроена в транслятор, и поэтому объявлять ее и описывать ее работу не требуется. Своим именем она просто вызывается на исполнение.

```
CALL init
CALL process
CALL result
```

В этих трех строках происходит вызов на исполнение (CALL) поочередно всех трех пользовательских процедур, объявленных выше.

```
END
```

На этом заканчивается текст основной программы, и мы переходим к описанию работы процедур. Каждое такое описание ограничено сверху и снизу служебными словами SUB и END SUB. Внутри же этих границ перечисляются те действия, которые должны быть выполнены в данной процедуре.

```
    SUB init
ret: INPUT "Сторона a ==>"; r(1)
    INPUT "Сторона b ==>"; r(2)
    INPUT "Сторона c ==>"; r(3)
```

Первые три оператора процедуры init предназначены для ввода с клавиатуры (INPUT) числовых значений первого, второго и третьего элементов массива r(1), r(2) и r(3). Этими значениями будут длины сторон треугольника. (Заметим, что первый оператор помечен — метка ret отделена от него двоеточием. Видимо, где-то в последующем тексте будет ссылка на эту строку.)

```
r(4) = r(1) : r(5) = r(2)
```

Для дальнейших действий, а именно для простоты организации цикла, нам потребуются еще два вспомогательных элемента массива r(4) и r(5). Им присваиваются те же значения, что и первым двум элементам.

Далее следует блок циклического оператора.

```
FOR i=1 TO 3
   (ТЕЛО ЦИКЛА)
NEXT i
```

Он ограничен сверху и снизу служебными словами FOR... и NEXT... Внутри этих границ расположено тело цикла. Для повышения наглядности его принято записывать с небольшим смещением вправо по отношению к границам.

```
IF r(i)>=r(i+1)+r(i+2) THEN
   (ТЕЛО УСЛОВНОГО ОПЕРАТОРА)
END IF
```

В данном случае, несмотря на то что запись тела цикла произведена посредством трех строк, оно состоит всего лишь из одного действия. Правда, это действие не простое, оно представляет собой целый блок — блок условного оператора, который в свою очередь имеет верхнюю и нижнюю границы (IF и END IF), а внутри них — свое собственное тело, которое может содержать в себе много разных действий, как простых, так и блочных.

В рассматриваемой программе тело условного оператора содержит два простых действия. Они будут выполняться одно за другим.

```
PRINT "Это не треугольник!"
GOTO ret
```

Сначала следует вывод (PRINT) сообщения на экран, а затем безусловный переход (GOTO) на ту строку процедуры, которая имеет метку ret (первый из трех операторов INPUT). Таким образом, ошибочные исходные данные могут быть заменены на верные при повторном вводе.

```
END SUB
```

Описание первой процедуры окончено.

```
SUB process
f=0
```

Первое действие внутри процедуры process предельно просто. Оно состоит в присваивании флаговой переменной f ее начального значения ноль. Если при дальнейшем выполнении процедуры оно не изменится, то именно его процедура передаст в основную программу для последующего анализа. (Вспомним, что данная переменная имеет статус SHARED.)

Далее следует блок циклического оператора.

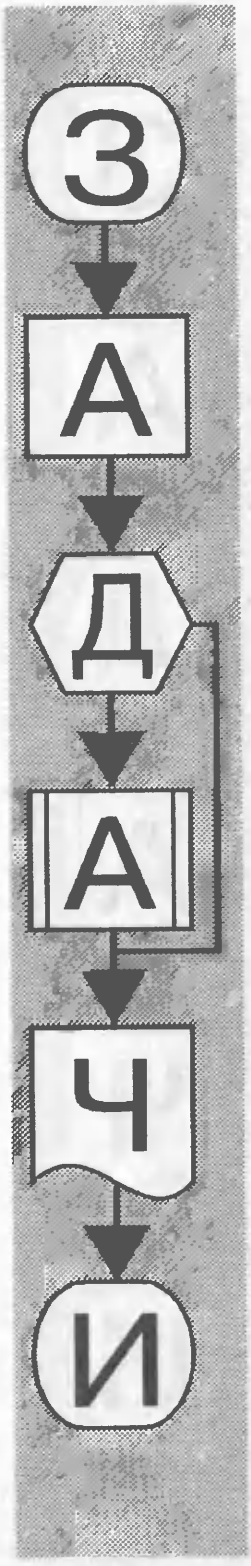
```
FOR i=1 TO 3
   (ТЕЛО ЦИКЛА)
NEXT i
```

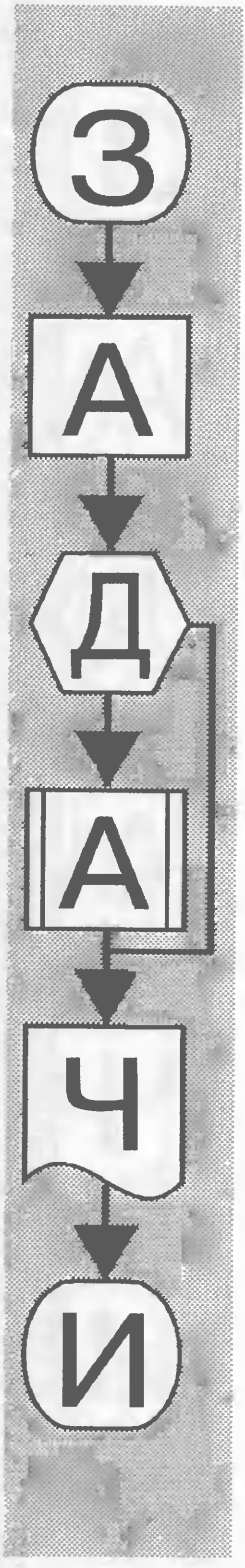
Тело цикла опять-таки, как и в процедуре init, содержит в себе один только блок условного оператора. Конечно, условие уже другое, но структура этого блока та же.

```
IF r(i)*r(i)=r(i+1)*r(i+1)+r(i+2)*r(i+2) THEN
   (ТЕЛО УСЛОВНОГО ОПЕРАТОРА)
END IF
```

(Необходимое замечание. Теорема Пифагора в условии записана необычным образом. Каждая сторона умножается сама на себя, вместо того чтобы возводиться в квадрат. Этому есть причина. Дело в том, что транслятор QBasic почему-то неправильно воспринимает обычную запись. Это можно проверить, выполнив отдельно следующий фрагмент:

```
if 5^2= 3^2+4^2 then print "Все в порядке" else print "Ошибка!")
```





Тело условного оператора содержит в себе два простых (не блочных) действия.

```
f=1
EXIT FOR
```

Флаговой переменной присваивается новое значение, сигнализирующее о том, что прямой угол в рассматриваемом треугольнике обнаружен. После этого дальнейший поиск становится бессмысленным, и мы от него отказываемся, прерывая выполнение цикла (EXIT FOR).

```
END SUB
```

На этом описание второй процедуры заканчивается, и мы переходим к последней, третьей.

```
SUB result
```

Процедура result состоит из трех действий: одного сложного (составного) и двух простых.

```
IF f=0 THEN PRINT "В треугольнике нет прямого угла!": EXIT SUB
```

Первое, представляя собой условный оператор, могло бы быть записано, как и в предыдущих процедурах, в блочном виде:

```
if f=0 then
   print "В треугольнике нет прямого угла!"
   exit sub
end if
```

Та форма записи в одну строку, что приведена в тексте, эквивалентна блочной. Выбор в ее пользу был сделан только лишь затем, чтобы продемонстрировать и такую возможность.

Два последующих простых действия

```
PRINT "Это прямоугольный треугольник!"
PRINT "Его гипотенуза — сторона "; r(i)
```

— это операторы вывода текстовых сообщений на экран.

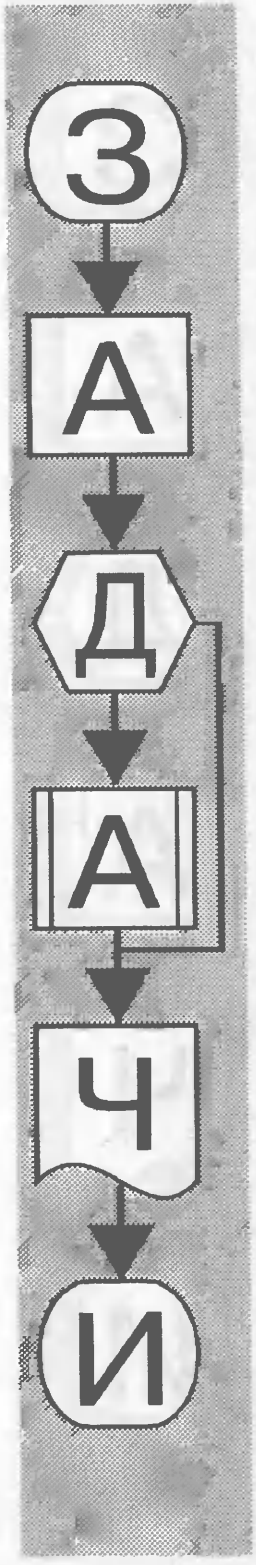
```
END SUB
```

Конец описания последней, третьей процедуры и одновременно конец всего текста программы.

Следует заметить, что в представленном варианте текста не все фрагменты являются обязательными. Так, например, работа программы не изменилась бы, если из текста декларативной части убрать все три строчки, в которых объявляются имена процедур. Также и в основном блоке можно было бы при вызове на исполнение процедур написать лишь их имена без служебного слова CALL. В этом отношении требования языка QBasic не очень строгие. Однако в некоторых ситуациях отступления от правил могут привести к ошибке в работе программы. Кроме того, в других языках программирования, как, например, в TurboPascal, эти требования к записи текста очень строги. Поэтому из соображений общепрограммистской дисциплины облегчать текст не рекомендуется.

Помимо этого, надо бы придерживаться еще и правил хороше-

го тона при написании программ и сопровождать текст ясными и по возможности полными комментариями. Это облегчит чтение программы незнакомому с ней человеку, а иногда даже и самому автору, особенно в том случае, когда он писал ее давно и уже успел кое-что позабыть.



Комментарии к тексту программы на TurboPascal

PROGRAM triangle;

Объявляется имя программы. Эта строка не является обязательной, но правила хорошего тона в программировании требуют, чтобы она была. Обычно в этой же строке (или в нескольких последующих) программист также указывает в качестве комментария краткое содержание программы. Комментарии в TurboPascal записываются в фигурных скобках.

USES CRT;

Объявляется имя одного (или нескольких) библиотечных модулей. Модуль — это набор некоторых готовых подпрограмм, который является приложением к транслятору. В частности, модуль CRT содержит в себе стандартные функции и процедуры работы с экраном монитора. Для того чтобы можно было использовать какие-либо из этих готовых процедур (а не писать их самому), нужно присоединить соответствующий модуль к транслятору. Именно это и выполняется посредством директивы USES.

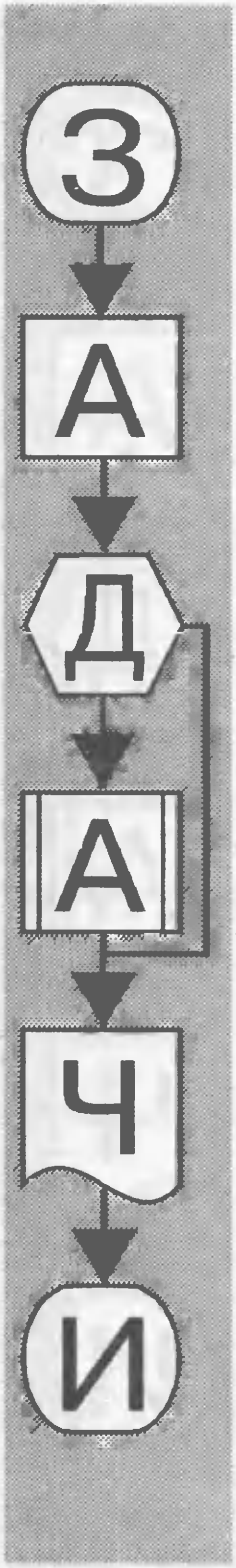
```
VAR r: ARRAY[1..5] OF REAL;
   i: BYTE;
   f: BOOLEAN;
```

Объявляются имена фигурирующих в программе глобальных, т. е. доступных для всех подпрограмм, переменных (VAR) с указанием их типов, причем имя от типа отделяется двоеточием. Тип переменной обязательно (в отличие от БЕЙСИКА) должен быть указан.

В данном случае речь идет о трех переменных. Одна из них, r, — это одномерный массив (ARRAY) из пяти элементов, причем указано, что элементами являются вещественные числа (REAL). Вторая переменная, i, — это целое число от 0 до 255 (BYTE). Наконец, f является переменной логического типа (BOOLEAN), т. е. может принимать всего лишь два значения: ИСТИНА (TRUE) или ЛОЖЬ (FALSE).

Далее идет блок, в котором объявляются пользовательские процедуры и функции. Но в отличие от QBasic, где имена процедур объявляются в начале, перед текстом основной программы, а описание их работы приводится после нее, в TurboPascal описание следует сразу же за объявлением имен. И только после этого пойдет текст основной программы.

```
PROCEDURE init;
LABEL ret;
BEGIN
   (ТЕЛО ПРОЦЕДУРЫ)
END;
```



Каждое такое описание ограничено сверху и снизу служебными словами BEGIN и END (эти слова очень часто применяются в любой программе, обозначая собой начало и конец какого-либо блока; для них даже принято особое название — операторные скобки). Внутри же этих границ перечисляются те действия, которые должны быть выполнены в описываемой процедуре.

В данном случае описывается процедура с именем init. Какая-то строка в ней будет, по-видимому, иметь метку (LABEL). Чтобы не принять ошибочно ее имя ret за переменную, она специально объявляется.

```
ret: WRITE ('Сторона a ==>'); READLN (r[1]);
     WRITE ('Сторона b ==>'); READLN (r[2]);
     WRITE ('Сторона c ==>'); READLN (r[3]);
```

Первые три строки предназначены для ввода с клавиатуры числовых значений первого, второго и третьего элементов массива r[1], r[2] и r[3]. Оператор ввода имеет две разновидности: READ и READLN. В первом случае курсор по окончании ввода остается в том месте экрана, где это окончание его застало. Во втором случае курсор перейдет на новую строку.

Для того чтобы ввод данных не был, как говорят, «слепым», предварительно выполняется вывод поясняющего текстового сообщения. Осуществляет вывод оператор WRITE, тоже имеющий разновидность WRITELN. В приведенном фрагменте после вывода поясняющего текста курсор останется в той же строке, а по окончании ввода перейдет на новую.

```
r[4]:=r[1]; r[5]:=r[2];
```

Двум вспомогательным элементам массива r[4] и r[5] присваиваются те же значения, которые имеют первые два элемента. Следует обратить внимание на то, что присваивание, в отличие от БЕЙСИКА, обозначается не просто знаком равенства, а с дополнительным двоеточием.

Далее следует блок циклического оператора (FOR...DO).

```
FOR i:=1 TO 3 DO
   (ТЕЛО ЦИКЛА);
```

В данном случае, несмотря на то что запись тела цикла произведена посредством пяти строк, оно состоит всего лишь из одного действия. Правда, это действие не простое, а представляет собой целый блок — блок условного оператора (IF...THEN).

```
IF r[i]>=r[i+1]+r[i+2] THEN
   (ТЕЛО УСЛОВНОГО ОПЕРАТОРА);
```

В рассматриваемой программе тело условного оператора содержит в себе два простых действия. В ПАСКАЛЕ действует одно важное правило. Если тело какого-нибудь оператора содержит в себе не одно, а несколько действий, то их надо заключать в операторные скобки BEGIN...END; в данном случае как раз это и сделано.

```
BEGIN
WRITELN "Это не треугольник!";
GOTO ret;
END;
```

Внутри тела условного оператора присутствует оператор безусловного перехода (GOTO) на объявленную выше метку.

```
PROCEDURE process;
BEGIN
f:=FALSE;
```

Первым действием в процедуре process является присваивание флаговой переменной f (логического, как было объявлено, типа) ее начального значения.

Далее следует оператор цикла, который, как и в предыдущей процедуре init, содержит в своем теле лишь одно действие, которое, согласно упомянутому выше правилу, не требует, чтобы его заключали в операторные скобки, поскольку оно одно.

```
FOR i:=1 TO 3 DO
```

Это единственное действие выражается условным оператором

```
IF SQR(r[i])=SQR(r[i+1])+SQR(r[i+2]) THEN
```

тело которого теперь обязано быть помещенным в операторные скобки,

```
BEGIN
f:=TRUE;
EXIT;
END;
```

поскольку в нем уже два действия.

В первом из них происходит присваивание флаговой переменной нового значения, означающего, что прямой угол в треугольнике найден. Второе действие (EXIT) приводит к прекращению дальнейшего поиска и выходу из блока условного оператора, а вместе с тем и к выходу из цикла.

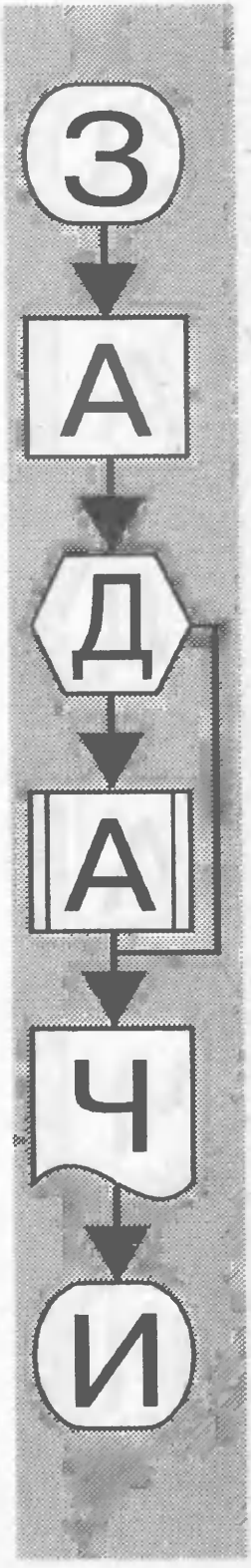
```
END;
```

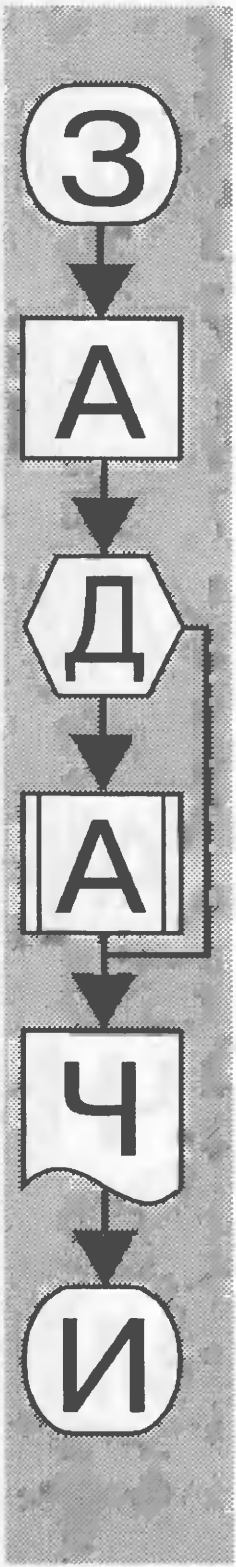
Более того, на этом завершается и вся процедура process.

```
PROCEDURE result;
BEGIN
IF NOT(f) THEN
   BEGIN
   WRITELN ('В треугольнике нет прямого угла!');
   EXIT;
   END;
WRITELN ('Это прямоугольный треугольник!');
WRITELN ('Его гипотенуза — сторона ',r[i]:3:1);
END;
```

Процедура result состоит из трех действий: одного сложного и двух простых. Сложное — это условный оператор, в теле которого, в свою очередь, два простых оператора. Структура процедуры легко просматривается благодаря горизонтальным сдвигам некоторых строк.

Следует дать пояснения ко второму оператору вывода. Он имеет две особенности. Во-первых, в списке вывода, который заключен в круглые скобки, через запятую перечислены два элемента, которые нужно вывести на экран: текстовое сообщение и





числовое значение переменной r[i]. Во-вторых, для переменной указан формат вывода, т. е. тот вид, который это числовое значение должно иметь (всего три значащих цифры, причем одна из них после запятой). Указывать формат, конечно, не всегда обязательно, однако в данном случае число, выводимое на экран, резало бы глаз своей, например, экспоненциальной формой записи.

На этом описательная часть программы закончена, и можно перейти к основному блоку. При беглом взгляде на него может возникнуть сомнение в том, что он основной, слишком он мал по размеру. Но он таков именно потому, что подготовительная часть, где описывались процедуры, была весьма обширной.

Собственно вся основная программа состоит из исполнения в нужном порядке подготовленных выше процедур.

```
BEGIN
ClrScr;
init;
process;
result;
END.
```

Первая процедура (ClrScr) берется уже готовой из заказанного в начале программы модуля CRT. Ее назначение — первоначальная очистка экрана перед выводом на него запросов и ответов.

Затем производится ввод исходных данных при помощи процедуры init, после чего посредством процедуры process осуществляется поиск, и, наконец, его результаты выводятся на экран, за что отвечает процедура result.

Последней строчкой программы является оператор END с обязательной завершающей точкой.

Примечание. Для того чтобы после выполнения программы результат ее работы остался на экране (до нажатия любой клавиши), часто используют один стандартный прием. Непосредственно перед последним служебным словом END. (с точкой) добавляют еще одну строку:

```
REPEAT UNTIL KeyPressed;
```

Это пустой, т. е. не содержащий никакого действия, цикл REPEAT UNTIL с постусловием KeyPressed, которое является стандартной функцией из модуля CRT. Эта функция имеет логическое значение FALSE до тех пор, пока не нажата никакая клавиша. Тем самым условие выхода из цикла не выполняется, и он без конца повторяется. При этом на экране ничего не меняется, поскольку цикл пустой.

Задания для самостоятельного решения

1. Изменить программу так, чтобы в ней определялся вид треугольника: если он не прямоугольный, то является ли он тупоугольным или остроугольным.

2. Изменить программу так, чтобы треугольник в ней исследовался на правильность: является ли он равнобедренным, равносторонним.

3. Дополнить программу процедурой определения периметра и площади треугольника.

4. Дополнить программу процедурой вывода изображения треугольника на графический экран.

5. Для заданного треугольника определить высоту, проведенную к самой длинной стороне.

6. Для заданного треугольника определить высоту, проведенную к самой короткой стороне.

7. Для заданного треугольника определить медиану, проведенную к самой длинной стороне.

8. Для заданного треугольника определить медиану, проведенную к самой короткой стороне.

9. Для заданного треугольника определить радиус вписанной в него окружности.

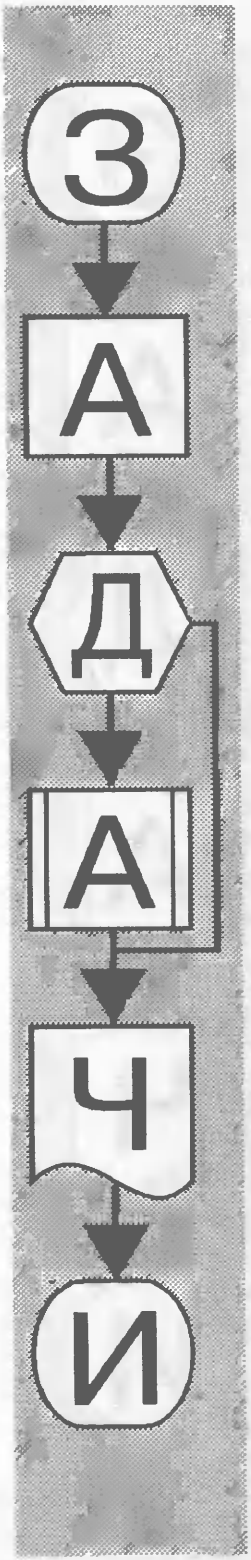
10. Для заданного треугольника определить радиус описанной вокруг него окружности.

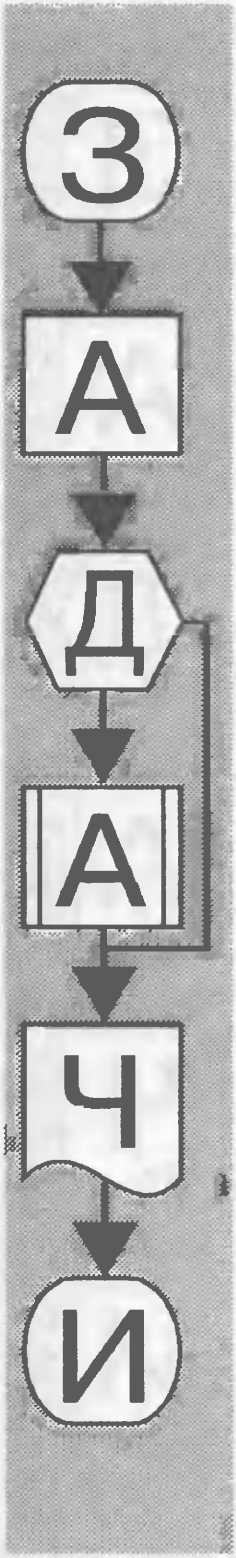
11. Для двух заданных треугольников определить, может ли один из них целиком поместиться внутри другого.

Исследовательское задание по теме «Треугольник»

Выяснить, как зависит площадь треугольника с заданным значением периметра от его формы.

Выяснить, как зависит площадь треугольника заданной формы от величины его периметра.





Ю. С. Варакин,
учитель средней школы № 550, Москва

ПРОЕКТЫ-ЗАДАНИЯ НА ЯЗЫКЕ ЛОГО ДЛЯ УЧАЩИХСЯ СТАРШИХ КЛАССОВ

В настоящее время все более популярным в качестве средства обучения основам компьютерного программирования в школе становится язык ЛОГО. Но, к сожалению, методической литературы по программированию в среде ЛОГО крайне мало, особенно на русском языке.

На основе опыта преподавания информатики в старших классах школы № 550 Москвы автором составлен сборник заданий, которые либо имеют интересный графический эффект при решении, либо представляют собой несложную компьютерную игру. Каждое задание сопровождается изложением его целей, формулировкой, рекомендациями к выполнению (неформальным описанием алгоритма), справочной информацией (списком примитивов языка ЛОГО, необходимых для составления программы) и примерным текстом программы, являющейся решением задания. В процессе решения этих заданий учащиеся знакомятся с разнообразными конкретными приемами программирования и алгоритмами, имеющими линейную, циклическую, ветвящуюся или рекурсивную структуры.

Как правило, для составления программы, являющейся решением задания, ученику достаточно заменить приведенные рекомендации, записанные в виде текста, на команды языка программирования, понятные компьютеру. По усмотрению преподавателя можно предложить учащимся условия задачи без использования подробных рекомендаций для ее решения.

В данной статье представлены три задания, иллюстрирующие такие различные задачи на программирование, как создание динамической графической композиции, компьютерного мультфильма и простой компьютерной игры.

Задание 1

Цели

1. Составление и отладка программ, использующих циклические алгоритмы и параметры, изменяющиеся от цикла к циклу.

2. Применение рекурсии для получения бесконечного выполнения программы.

3. Получение оригинальных графических композиций и выделение наиболее удачных.

Формулировка

Составить программу, в результате работы которой формируется композиция, составленная из геометрических фигур, размеры которых увеличиваются от цикла к циклу, при этом геометрическая фигура как бы растет в одну сторону. Если в каждом цикле

геометрическую фигуру не только увеличивать в размере, но и поворачивать на некоторый угол, то сформируется графическая композиция, зависящая от угла поворота (рис. 1).

Необходимо просмотреть различные варианты рисунков и определить углы поворота геометрической фигуры для получения наиболее интересной композиции.

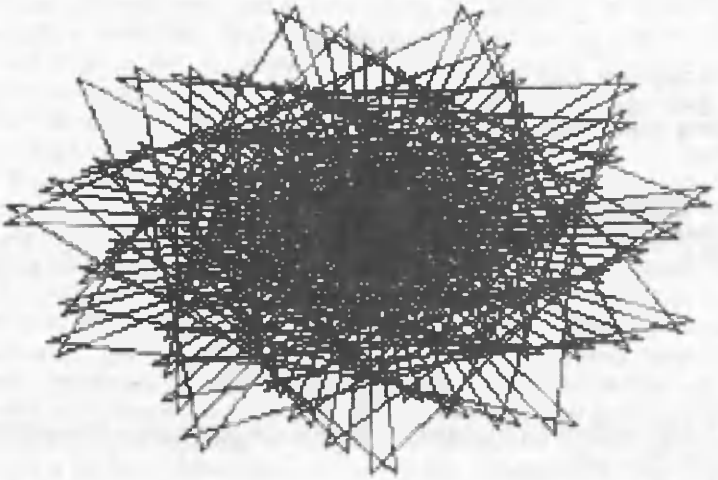


Рис. 1

Рекомендации к выполнению

Программу для выполнения задания можно разделить на две процедуры.

Первая — процедура изображения геометрической фигуры, например многолучевой звезды, полученной повторением движения черепашки вперед с поворотом на определенный угол.

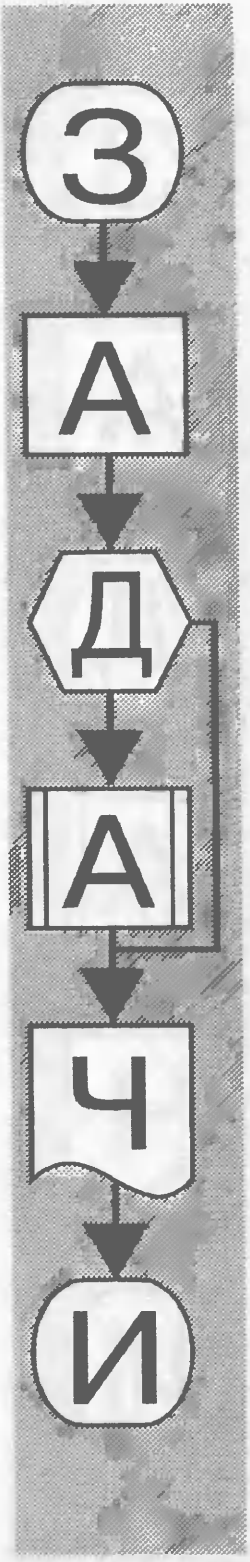
Во второй процедуре необходимо:

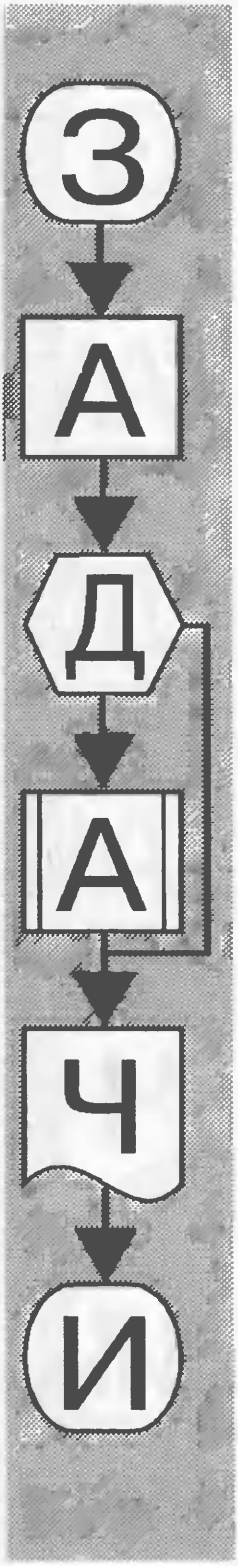
- задать переменную, определяющую размер геометрической фигуры (начальное значение переменной можно принять нулевым);
- очистить экран монитора от предыдущей графики;
- установить цвет контура рисунка;
- установить цвет экрана (он может меняться, т. е. быть случайным при каждом новом выполнении процедуры).

Далее в процедуре выполняется ряд циклов изображения геометрической фигуры с увеличением ее размера и поворотом на некоторый угол от цикла к циклу. После выполнения некоторого числа циклов производится распечатка угла поворота геометрической фигуры для выделения наиболее интересных композиций и рисунок повторяется с новым значением угла поворота.

Используемые примитивы языка ЛОГО

ht	— спрячь черепашку
fd	— вперед
rt	— направо
cg	— сотри графику
pd	— перо опусти





setc — установи цвет пера
setbg — установи цвет экрана
name — имя локальной переменной
repeat — повтори
pr — распечатай
random — случайно

Программа

```
to морские_ежи :g :n
rg ht name 0 "n pd repeat 150 [ звезда :n
rt :g name :n + 1 "n] pr :g морские_ежи :g + 1 :n
end

to звезда :n
repeat 5 [fd :n rt 144]
end
```

Задание 2

Цели

1. Составление и отладка программ для получения простейших мультипликаций.

2. Обретение навыков работы с листом форм и с несколькими черепашками.

Формулировка

Составить программу для компьютерного мультфильма по следующему сценарию.

Слева в верхней части экрана монитора появляется движущийся трактор, который подъезжает к полю и засевает его саженцами разноцветных цветов (рис. 2). После окончания работы трактор уезжает с поля и исчезает у правой границы экрана монитора.

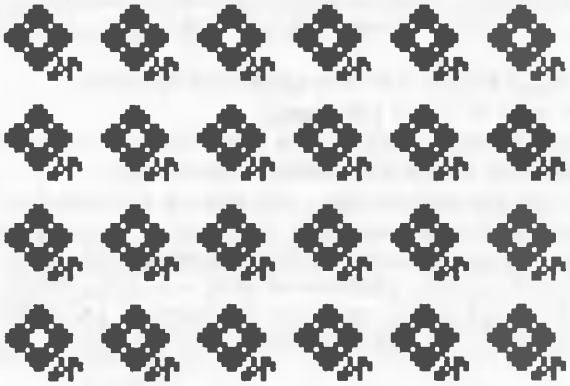


Рис. 2

Рекомендации к выполнению

Для выполнения предложенного задания можно использовать двух черепашек. Одна черепашка, приняв форму трактора, «подъезжает» к полю в центральной части экрана и совершает по нему челночные движения, создавая эффект посадки саженцев. Вторая черепашка, став невидимой, выполняет вслед за первой черепашкой аналогичные передвижения по полю, оставляя за собой изображения саженцев цветов с помощью команды «штамп». Затем первая черепашка «уезжает» с поля и исчезает у правого края монитора, став невидимой.

Программу для выполнения задания можно представить двумя процедурами.

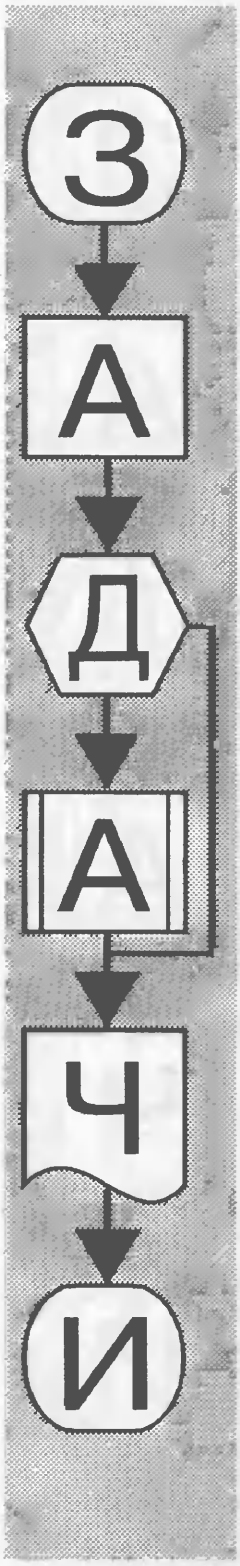
В первой процедуре сначала выполняется очистка экрана, установка черепашек в исходное состояние. Далее первая черепашка принимает форму трактора, встает слева в верхней части экрана монитора, становится видимой, поворачивается направо и начинает движение. Для уменьшения скорости движения продвижение черепашки вперед чередуется командами звукового сопровождения. Движение черепашки завершается установкой ее в начальную точку поля, которое необходимо засеять цветами. Затем вторая черепашка, оставаясь невидимой, принимает форму цветка, поворачивается направо и занимает позицию на экране монитора немного позади трактора. После этого начинается совместное движение двух черепашек по полю.

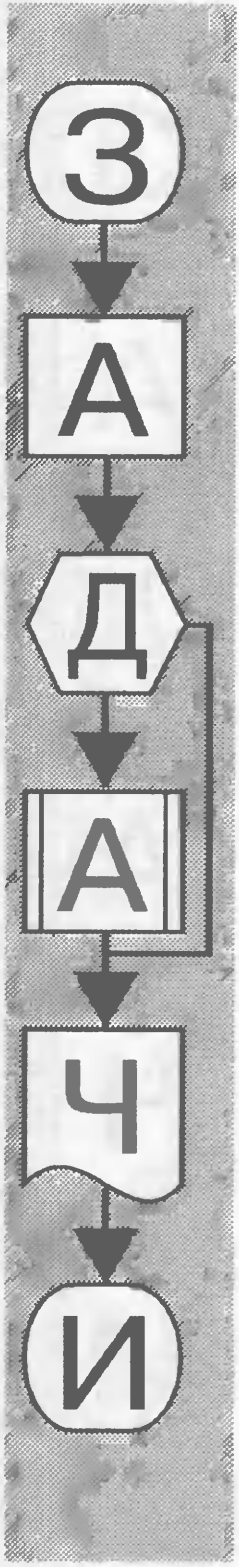
Движение черепашек по полю представляет собой повторение процедур рисования грядок с цветами. После того как нарисована первая грядка, трактор должен развернуться назад для рисования следующей грядки. При этом форма черепашки, изображающей трактор, должна поменяться на зеркальную, так как при повороте черепашки ее форма не поворачивается. Затем черепашка устанавливается таким образом, чтобы первая (изображающая трактор) следовала за второй (изображающей цветок). После завершения посадки черепашка, изображающая трактор, «уезжает» с поля.

Процедура для изображения грядки представляет собой повторение команд продвижения вперед первой и второй черепашек, при этом вторая черепашка, оставаясь невидимой, выполняет команду «штамп», оставляя на грядке изображение цветов, при этом ее перо должно быть опущено.

Используемые примитивы языка ЛОГО

cg	— сотри графику
pu	— перо подними
pd	— перо опусти
ht	— спрячь черепашку
st	— покажи черепашку
home	— домой (занять исходное место)
tell	— обращение к указанной черепашке
setsh	— установи форму
set	— установи цвет
setpos	— установи черепашку в точку с указанными координатами X и Y
rt	— направо
lt	— налево
repeat	— повтори
random	— случайный





fd	— прямо
wait	— жди
stamp	— штампуй свою форму
tone	— звук

Программа

```
to поле
cg tell [0 1] pu home tell 0 setsh 28 setpos [-170 90]
st rt 90 repeat 20 [fd 5 tone 50 1]
tell 1 rt 90 setsh 24 ht setpos [-120 90] repeat 2 [грядка
tell 0 setsh 1 rt 90 fd 60 rt 90 fd 40 wait 30
tell 1 fd 40 rt 90 fd 60 rt 90
грядка
tell 0 setsh 28 lt 90 fd 60 lt 90 fd 40 wait 30
tell 1 fd 40 lt 90 fd 60 lt 90] tell 0 repeat 60 [fd 5 tone 50 1] ht
end

to грядка
repeat 6 [tell 0 repeat 8 [fd 5 tone 50 1]
tell 1 fd 40 pd setc 2 + random 14 stamp pu wait 5]
end
```

Задание 3

Цели

1. Составление и отладка программ, использующих клавиатуру компьютера для управления черепашкой.

2. Знакомство с командой ввода информации readchar, датчика клавиши key и датчика цвета экрана colorunder.

Формулировка

Составить программу для компьютерной игры по следующему сценарию.

Между черепашкой, находящейся у левого края монитора, и правым краем экрана монитора, где расположен условный берег моря, случайным образом расставлены деревья (рис. 3).

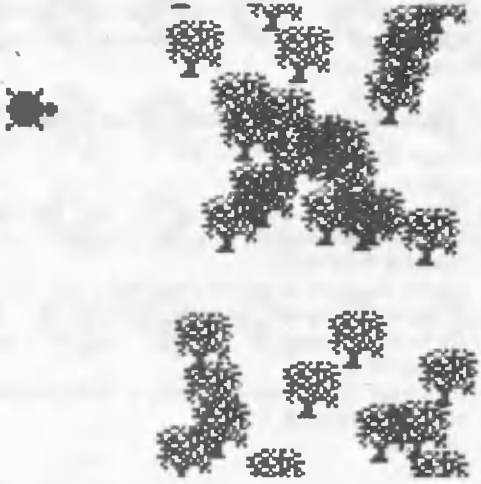
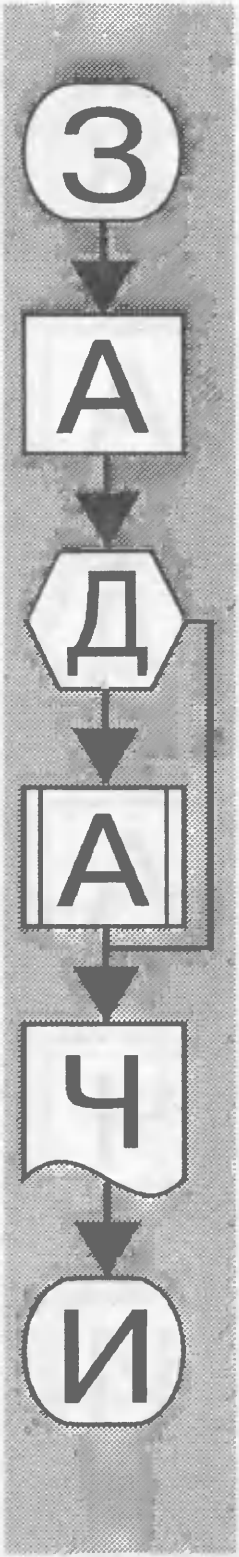


Рис. 3

После запуска программы черепашка начинает движение в направлении леса и может поворачивать направо и налево при нажатии соответствующей клавиши. Задача играющего — провести черепашку к берегу моря так, чтобы она не столкнулась с деревом. Если она сталкивается с деревом, то выполнение процедуры прекращается и цвет экрана становится оранжевым. Если черепашка достигает правого края экрана монитора, то она также останавливается, но экран монитора окрашивается в голубой цвет.



Рекомендации к выполнению

Программу для выполнения задания можно разделить на три процедуры.

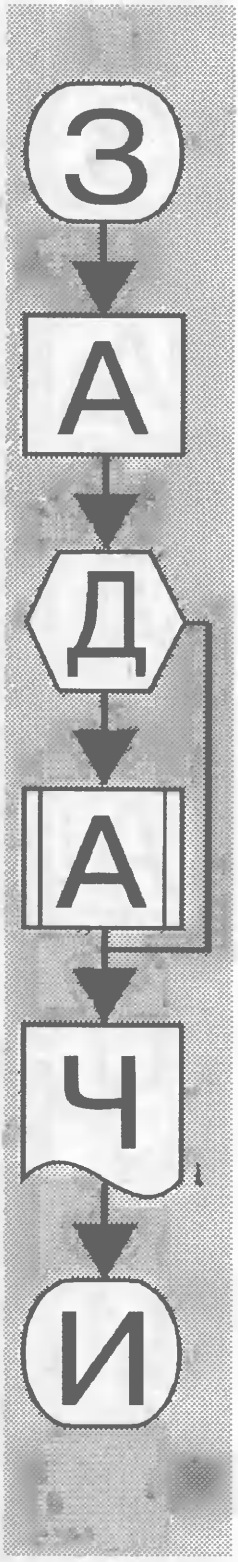
Первая процедура выполняет начальную установку, при которой экран очищается от предыдущей графики, цвет экрана становится нейтральным, черепашка становится невидимой, изображает условный берег моря и принимает форму дерева, которая затем с помощью команды «штамп» отштамповывается случайным образом в центральной части монитора, создавая изображение леса. Затем черепашка принимает свою привычную форму, встает в левой части экрана монитора, принимает необходимый курс и управление передается второй процедуре, которая приводит черепашку в движение и позволяет управлять ею с помощью клавиш клавиатуры (например, клавиш Q и P).

Во второй процедуре черепашка движется вперед, пока не будет нажата клавиша для поворота черепашки направо или клавиша для поворота черепашки налево. В этом случае управление передается третьей процедуре, имеющей параметр, по которому в зависимости от нажатой клавиши выполняется поворот направо или налево на небольшой угол.

В процессе управления черепашкой возможны два варианта: либо оператор, управляющий черепашкой, находит путь через лес и проводит черепашку к правому краю монитора, тогда цвет экрана становится голубым и черепашка останавливается; либо черепашка «наезжает» на препятствие, что определяется с помощью команды датчика цвета экрана под центром черепашки. В этом случае цвет экрана становится оранжевым, и черепашка останавливается.

Используемые примитивы языка ЛОГО

cg	— сотри графику
if	— если
wait	— жди
ht	— спрячь черепашку
key?	— клавиша
st	— покажи черепашку
readchar	— команда ввода символа клавиши
pu	— перо подними
pd	— перо опусти
setx	— перемести черепашку по оси X
sety	— перемести черепашку по оси Y
lt	— налево
rt	— направо
setsh	— установи форму
stamp	— отштампуй форму
seth	— установи курс
stop	— стоп



xcor — значение координаты X
ycor — значение координаты Y
setbg — установи цвет экрана
name — имя локальной переменной
repeat — повтори
random — случайный
colorunder — команда-датчик цвета экрана под центром черепашки

Программа

```
to игра
rg pu ht setx 155 pd fd 500 pu setsh 23
name 0 "a repeat 30 [setx -130 + :a name :a + 5 "a
sety random 350 pd stamp pu]
setsh 0 st setx -230 sety random 150 seth 90 движение
end

to движение
fd 3 if key? [управление readchar]
if x > 150 [setbg 10 stop]
if colorunder = 1 [setbg 14 stop] wait 3 движение
end

to управление :x
if :x = "q [lt 5]
if :x = "p [rt 5]
end
```

Е. В. Утлинский, Э. Г. Аввакумов,
Институт общеобразовательной школы Российской академии образования, лаборатория информатики

ТЕЛЕКОММУНИКАЦИОННАЯ СЕТЬ TIT-BIT*

Итак, уважаемый читатель, надеемся, что ваше аппаратное и программное обеспечение подключено и настроено для сеанса связи и вы умеете производить дозвон до телекоммуникационного сервера. В данной статье мы рассмотрим, как могли бы выглядеть первые сеансы связи некоторого начинающего пользователя компьютерных сетей.

Сеанс первый (гость первый раз входит на сервер)

Рассмотрим телекоммуникационный сеанс связи с одним из серверов образовательной сети TIT-BIT, расположенным в Москве и имеющим телефонный номер (095) 924-7864. Используя телекоммуникационную программу (ПО TIT-BIT или другую), звоним на сервер. После того как установлена нормальная связь с сервером, вы увидите на экране его заставку:

Добро пожаловать в телекоммуникационую сеть TIT-BIT!
Модем: (095) 924-7864 19200/V.42bis/MNP5 **станция работает круглосуточно**
Операторская служба: (095) 923-01-61
электронная почта **телеконференции** **новости образования, науки, бизнеса** **реклама Ваших разработок**

Гостевой идентификатор: TIT-BIT
Подсказка по команде:HELP

Обычно заставка содержит название сервера, часы его работы, характеристики установленных на сервере модемов, имя оператора и другие полезные данные. В этот момент полезно включить режим записи в файл выводимой на экран информации (в ПО

---

* Продолжение. Начало см.: Информатика и образование. 1994. № 4, 6.

TIT-BIT это делается с помощью нажатия клавиши <F5>). Не следует звонить на сервер в его нерабочее время. В это время «модемные» телефонные номера могут использоваться для других целей.

Пока вы видите только заставку сервера. Чтобы получить возможность работать с сервером, необходимо ввести свой идентификатор (последовательность из 2—8 латинских символов и/или цифр).

Все пользователи телекоммуникационного сервера делятся на три категории: гости, пользователи и абоненты. Пользователи и абоненты имеют свои идентификаторы входа на сервер и пароли. Гости могут входить на сервер под идентификатором TIT-BIT. Поскольку вы не имеете своего идентификатора и пароля, войдите на сервер под гостевым идентификатором:

Идентификатор: TIT-BIT

Появляется следующая заставка:

Вход открыт. Длительность сеанса (минут): 10

Уважаемый гость!
Добро пожаловать на сервер образовательной сети TIT-BIT. Сеть объединяет высшие и средние учебные заведения России, учреждения управления образованием и одиночных пользователей.
О том, как повысить уровень доступа до пользователя, смотрите в разделе \INFO\README. О том, как стать абонентом, смотрите в разделе \INFO\REGISTR.
Нажмите <ENTER>

Итак, вы вошли на сервер и вам разрешено работать с ним в течение 10 минут. Все серверы образовательной сети TIT-BIT имеют примерно одинаковую организацию. Войдя на сервер, вы попадаете как бы в дисковую операционную систему (например, на логический диск F):

F:\>

Единственная команда, известная вам в данный момент, — это HELP — получение подсказки (она была показана на заставке сервера). Воспользуемся ею:

F:\> HELP

Система команд телекоммуникационного сервера	
HELP	**..... вызов подсказки**
DIR	**..... просмотр содержимого раздела**
CD	**..... смена текущего раздела**
TYPE	**..... просмотр содержимого текстовых файлов**
DEL	**..... удаление файлов**
REN	**..... переименование файлов**
HOME	**..... переход в личный раздел**
ACCESS	**..... показ разрешенных команд**
SZ	**..... прием любых файлов с сервера**
RZ	**..... отправка любых файлов на сервер**
SMAIL	**..... прием почтовых файлов с сервера**
RMAIL	**..... отправка почтовых файлов на сервер**
STOP	**..... завершение сеанса связи**
Команда HELP с параметром выдает подробное описание команды, например: HELP DIR, HELP SMAIL и т. д.	

Здесь перечислены все команды, которые может выполнять сервер. Некоторые из этих команд являются полными аналогами команд операционной системы: DIR, CD, TYPE, REN, DEL. Посмотрим описание команды DIR:

```
F:\> HELP DIR

Синтаксис:  DIR [Имя_Файла] или DIR [Имя_Раздела]
            параметр команды может быть опущен
Описание:   предназначена для просмотра содержимого разделов
Примеры:    DIR — просмотр текущего раздела
            DIR *.TXT — поиск файлов с любыми именами
            и расширением TXT в текущем разделе
```

Дайте команду DIR, чтобы посмотреть содержание текущего раздела (сейчас текущим разделом является корневой раздел сервера):

```
F:\> DIR

  FILE        [Раздел]   Файловый раздел
  INFO        [Раздел]   Информационный раздел
  GUEST       [Раздел]   Гостевой раздел
  MAIL        [Раздел]   Почтовый раздел
  CONF        [Раздел]   Раздел телеконференций
```

Данный сервер имеет пять глобальных разделов (каталогов). Раздел FILE служит для хранения архивов программ и документации. Давайте войдем в этот раздел с помощью команды CD и посмотрим его содержание с помощью команды DIR:

```
F:\> CD FILE

F:\FILE> DIR

  ANTIVIR     [Раздел]   Антивирусы
  EDUCAT      [Раздел]   Обучающие программы
  STANDARD    [Раздел]   Стандарты на среднее образование
  STUDENT     [Раздел]   Студенческие разработки
  UKNC        [Раздел]   Программы для УКНЦ
```

Допустим, вас заинтересовали программы для УКНЦ. Вы можете войти в раздел UKNC с помощью команды CD и посмотреть его содержание с помощью команды DIR:

```
F:\FILE> CD UKNC

F:\FILE\UKNC> DIR

  about              438   Содержание раздела FILE\UKNC
  genereng    .arj   95889
  learneng    .arj   39465
  loto        .arj   25120
```

Этот раздел уже не содержит подразделов, а содержит только файлы. Как правило, в каждом файловом разделе есть текстовый файл about, информирующий о содержании текущего раздела. Выведите текст этого файла на экран с помощью команды TYPE:

```
F:\FILE\UKNC> TYPE about

Содержание раздела FILE\UKNC
В этом разделе свои разработки для УКНЦ представляет центр
НИТ Ярославского института повышения квалификации учителей

  genereng    .arj   Пакет «General English»
  learneng    .arj   Пакет «Learn English»
  loto        .arj   Пакет «Лото»
```

Попробуйте с помощью команды REN переименовать какой-либо файл из данного раздела:

F:\FILE\UKNC> REN loto.arj aaa.txt

Не хватает прав доступа

Попробуйте с помощью команды DEL удалить какой-либо файл из данного раздела:

F:\FILE\UKNC> DEL loto.arj

Не хватает прав доступа

Вам не разрешили выполнить эти действия, поскольку существуют определенные права доступа в каждый раздел. Пользователи и абоненты имеют больше прав доступа, гостей же в некоторые разделы могут просто не пустить. Находясь в каком-либо разделе, вы можете посмотреть свои права доступа в этот раздел с помощью команды ACCESS:

F:\FILE\UKNC> ACCESS

Доступные команды: CD, DIR, TYPE, SZ

Новая команда — SZ. Она предназначена для приема файлов с телекоммуникационного сервера на ваш компьютер. Пересылка файлов между двумя компьютерами осуществляется с помощью протокола передачи файлов — программно реализованного набора правил, определяющих последовательность пересылки файла и процедуры исправления неверно принятых данных.

Существует множество протоколов передачи файлов: Xmodem, Ymodem, Kermit и т. д. В настоящее время наиболее распространенным и эффективным является протокол передачи файлов Zmodem. Данный протокол имеет возможность рестарта при пересылке файла. Например, если при пересылке файла длиной 300 Кб из-за помех на телефонной линии на 299 Кб произошел обрыв связи, то при использовании старых протоколов передачи данных вам придется все начинать сначала, а при использовании протокола Zmodem вам останется принять еще 1 Кб. При пересылке файлов серверы сети TIT-BIT используют наиболее эффективный протокол Zmodem, ваша телекоммуникационная программа также должна поддерживать этот протокол.

Давайте с помощью команды SZ примем с сервера файл loto.arj:

F:\FILE\UKNC> SZ loto.arj

Сервер готов к пересылке файлов протоколом Zmodem

Если вы используете программу ПО TIT-BIT, то пересылка файла начнется сразу. Другим же телекоммуникационным программам может потребоваться дать команду приема файлов Download (обычно с помощью клавиши <PgDn>) и указать протокол передачи Zmodem, после чего начнется пересылка файла. Во время пересылки файла вы будете видеть на экране таблицу, сообщающую о параметрах и полноте производимой пересылки:

Прием

Файл	**LOTO.ARJ**	**Протокол**	**Zmodem CRC16**
Размер	**25120**	**Ошибки**	**0**
Передано	**8632**	**Время**	**00:01:24**
Блок	**512**	**Скорость**	**250**

32% ♦♦♦♦♦♦♦♦♦♦ . . . . . .

Указанный файл будет принят в каталог приема вашей телекоммуникационной программы (для ПО TIT-BIT это каталог \TIT-BIT\RECEIVE). Пересылка файла на скоро-

сти 2400 бит/с заняла примерно 4 минуты, а вас пустили в систему всего на 10 минут. Но этого же недостаточно для нормальной работы с сервером. Как же повысить свой уровень доступа? Для этого необходимо получить информацию из раздела \INFO\README (это было написано в заставке сервера).

Переходим в раздел \INFO\README и просматриваем его содержание:

F:\> CD \INFO\README

F:\INFO\README> DIR

access	**.txt**	**1435**	**Как увеличить уровень доступа**
rules	**.txt**	**2654**	**Правила поведения на сервере**
suppor	**.txt**	**2375**	**Предоставляемые услуги**

Информация, содержащаяся в данном разделе, очень важна. Поэтому примем все файлы из этого раздела на наш компьютер:

F:\INFO\README> SZ *.*

Сервер готов к пересылке файлов протоколом Zmodem

Как и в предыдущий раз, если вы используете программу ПО TIT-BIT, пересылка файлов начнется сразу, другим же телекоммуникационным программам может потребоваться дать команду приема файлов и указать на протокол Zmodem.

А сейчас давайте попробуем... Что за надпись появилась на экране?

Разъединение. Длительность сеанса (минут): 10

Сеанс второй (гость хочет стать пользователем)

Итак, вы получили информацию из раздела \INFO\README. В файле access.txt, в частности, объясняется, как повысить свой уровень доступа в систему. Для этого в раздел \GUEST сервера вам следует отправить файл, в котором сообщить о себе: имя и фамилию, род занятий, что вас интересует на сервере, номер вашего «модемного» телефона, идентификатор (2—8 символов) и пароль (6—20 символов), с которыми вы хотели бы входить на сервер.

Допустим, вас зовут Евгений Смирнов. Ваше письмо могло бы выглядеть так: «Здравствуйте. Меня зовут Евгений Смирнов, я декан физико-математического факультета Московского института связи. Меня интересуют новости образования. Номер моего модема: (095) 222-15-95. Я хотел бы входить на сервер под идентификатором SMIRN, с паролем DELTA-M».

После этого файл с письмом (например, letter.txt) следует поместить в каталог отправки вашей телекоммуникационной программы (для ПО TIT-BIT это каталог \TIT-BIT\SEND). Звоним на сервер и входим под гостевым идентификатором TIT-BIT. После этого переходим в раздел \GUEST:

F:\> CD \GUEST

Для отправки файла на сервер следует воспользоваться командой RZ:

F:\GUEST> RZ

Сервер готов к приему файлов протоколом Zmodem

Если вы используете программу ПО TIT-BIT, то вам будет необходимо ввести только имя отправляемого файла (letter.txt). Другим же телекоммуникационным программам может потребоваться дать команду отправки файлов Upload (обычно с помощью клавиши <PgUp>), указать протокол передачи Zmodem и имя пересылаемого файла, после чего начнется пересылка файла. Во время отправки файла вы будете видеть на экране таблицу, подобную той, которая выводится на экран при приеме файла. После отправки файла можно прекратить сеанс связи с помощью команды STOP:

F:\GUEST> STOP

Разъединение. Длительность сеанса (минут): 4

Эта команда позволяет вам закончить сеанс связи с сервером. Никогда не обрывайте связь, не выполнив этой команды. Иначе серверу может потребоваться некоторое время, чтобы понять, что вы ушли не попрощавшись. Во время этого вынужденного перерыва пользователи не смогут получить доступ в систему.

В течение суток ваше письмо прочитает оператор сервера, и на следующий день вы сможете войти в систему со своим идентификатором и паролем.

Сеанс третий (пользователь получает почтовый ящик)

На следующий день вы можете позвонить на сервер и попытаться войти под своим идентификатором:

Идентификатор: SMIRN

После ввода идентификатора система просматривает список пользователей и, если пользователь с данным идентификатором обнаружен, запрашивает пароль:

Пароль: *****

Каждый символ вводимого пароля отображается на экране как знак «*». Если пароль введен неправильно, система попросит вас еще раз ввести идентификатор и пароль. Если вы ошибетесь несколько раз при вводе пароля, произойдет рассоединение. Допустим, вы правильно ввели пароль и вошли на сервер. Появляется приветственная заставка:

Длительность сеанса (минут): 20

Евгений Смирнов (декан ФМФ МИС)
Ваш почтовый ящик: \MAIL\SMIRN

Вам повысили длительность сеанса до 20 минут, и вы стали обладателем почтового ящика. Почтовый ящик — индивидуальный раздел (каталог) пользователя на сервере, который позволяет получить доступ к системе локальной почты внутри сервера (а абоненты через почтовый ящик получают доступ к системе всемирной электронной почты, но об этом позже).

Почтовые ящики пользователей расположены внутри раздела \MAIL, и права доступа в каждый почтовый ящик определены таким образом, что читать оттуда может только его владелец, а писать туда могут все. Кроме того, существует специальный почтовый ящик \MAIL\POSTMAST, который принадлежит постмастеру (оператору сервера), куда можно направлять свои жалобы, предложения и пожелания.

Суть системы локальной электронной почты внутри сервера состоит в том, что вы можете оставлять свои сообщения в любом существующем почтовом ящике и все пользователи сервера могут оставлять сообщения в вашем почтовом ящике.

Допустим, в каталоге отправки вашей телекоммуникационной программы вы имеете файл letter.txt, который хотели бы отправить в ящик постмастера (оператора сервера). Для этого надо войти в его почтовый ящик с помощью команды CD и воспользоваться командой RZ:

F:\> CD \MAIL\POSTMAST

F:\MAIL\POSTMAST> RZ

Сервер готов к приему файлов протоколом Zmodem

Если вы используете программу ПО TIT-BIT, то вам будет необходимо ввести только имя отправляемого файла. Другим же телекоммуникационным программам может потребоваться дать команду отправки файлов Upload (обычно с помощью клавиши

<PgUp>), указать протокол передачи Zmodem и имя пересылаемого файла, после чего начнется пересылка файла.

Посмотрим, а не оставил ли кто-нибудь сообщения в вашем почтовом ящике. Для перехода в свой почтовый ящик можно воспользоваться командой HOME:

```
F:\MAIL\POSTMAST> HOME

F:\MAIL\SMIRN> DIR

        hello.txt   254   Здравствуйте, Евгений. Рады видеть Вас среди
```

В вашем почтовом ящике находится файл hello.txt. Выведем его содержимое на экран с помощью команды TYPE:

```
F:\MAIL\SMIRN> TYPE hello.txt

Здравствуйте, Евгений. Рады видеть Вас среди наших пользователей. Уже
сейчас Вам доступны файловые области, локальная почта внутри станции,
новости образования. Став абонентом (после заключения договора на об-
служивание), Вы сможете получить доступ к системе всемирной электрон-
ной почты. Информацию о заключении договора смотрите в разделе
INFO\REGISTR.
                                        С уважением, оператор сервера
```

Вы прочитали это послание. В будущем оно, вероятно, вам не понадобится, и его можно смело удалить, ведь почтовый ящик \MAIL\SMIRN принадлежит вам, и вы вольны делать в нем все, что угодно:

```
F:\MAIL\SMIRN> DEL *.*
Удалено файлов: 1
```

Система локальной электронной почты позволяет вам общаться с другими пользователями внутри сервера. А что делать, если вы хотите общаться с вашими коллегами, находящимися в другом городе или за пределами России? Для этого предназначена система всемирной электронной почты. Но ею могут пользоваться только абоненты. Информацию о том, как можно стать абонентом, получим в разделе \INFO\REGISTR.

```
F:\MAIL\SMIRN> CD \INFO\REGISTR

F:\INFO\REGISTR> DIR

  registr.txt  13200   Договор на обслуживание
  price.txt    9764    Расценки на информационные услуги

F:\INFO\REGISTR> SZ *.*

Сервер готов к пересылке файлов протоколом Zmodem

F:\INFO\REGISTR> STOP

Разъединение. Длительность сеанса (минут): 14
```

Сеанс четвертый (абонент получает электронный адрес)

Для заключения договора на информационное обслуживание следует внимательно ознакомиться с полученной информацией и позвонить на сервер по «голосовому» телефону. Оператор сервера подробно объяснит вам детали заключения договора.

Став абонентом после заключения договора на обслуживание, при входе на сервер вы увидите заставку:

```
Длительность сеанса (минут): 30

Евгений Смирнов (декан ФМФ МИС)
Ваш почтовый ящик:     \MAIL\SMIRN
Ваш электронный адрес: smirn@tlt-blt.msk.su
```

Вам повысили доступ до 30 минут и вы получили электронный адрес, что означает возможность пользоваться системой всемирной электронной почты. Электронный адрес имеет следующий смысл (читаем справа налево):

su **сервер расположен в стране с идентификатором su (Soviet Union). Россия продолжает использовать этот идентификатор и постепенно переходит к использованию идентификатора ru (Russia);**

msk **сервер расположен в Москве. Обычно название города записывается полностью. msk – одно из редких исключений из правила;**

tit-bit **название сервера;**

smirn **идентификатор абонента, зарегистрированного на этом сервере.**

Вот несколько примеров идентификаторов стран: su (Россия), ua (Украина), by (Белоруссия), gg (Грузия), us (США), uk (Великобритания), ge (Германия), fr (Франция). Примеры написания названий городов: yaroslavl (Ярославль), arkhangelsk (Архангельск), omsk (Омск). Исключения из правила: msk (Москва), spb (Санкт-Петербург), e-burg (Екатеринбург).

Таким образом, электронный адрес некоторого абонента, находящегося в Киеве (Украина), мог бы выглядеть так: sergey@institute.kiev.ua, а электронный адрес абонента, находящегося в Берлине (Германия), мог бы иметь такой вид: elsa@lab.berlin.ge.

Система электронной почты позволяет отправлять и принимать электронные письма. Это очень быстрый и надежный вид связи. Отправленное по электронной почте письмо идет в любую точку мира в среднем не дольше 6 часов. Электронное письмо — это обычный текстовый файл, снабженный «конвертом». Вот пример электронного письма ректору Омского университета (электронный адрес вымышлен):

---

to: rector@univer.omsk.su
from: smirn@tit-bit.msk.su
subject: приглашение

Здравствуйте. Приглашаю Вас посетить Московский институт связи.

С уважением, Евгений Смирнов

Здесь в поле «to:» указывается электронный адрес получателя письма, в поле «from:» указывается электронный адрес отправителя письма, в поле «subject:» указывается краткая тема письма. Далее, через пустую строку, следует текст письма. Вообще говоря, обязательным является только поле «to:» (поле «subject:» можно опускать, а поле «from:» допишется автоматически при отправке письма). Обратите внимание, что в служебных полях конверта письма после двоеточия обязательно должен быть один пробел. Отправляя письма внутри бывшего Советского Союза (за исключением прибалтийских государств), вы можете использовать русский язык, вся же международная переписка ведется только на английском языке, и использование символов национальных алфавитов там недопустимо.

Итак, вы научились писать электронные письма, теперь надо научиться их отправлять. Напомним, что такое почтовый ящик для абонента. Почтовый ящик — это раздел (каталог) сервера, через который абонент осуществляет свое взаимодействие с внешним миром, т. е. в свой почтовый ящик абонент отправляет исходящую корреспонденцию и из него же принимает входящую корреспонденцию. Отправкой и приемом почты занимается специальная программа, работающая на сервере, которая время от времени просматривает почтовые ящики абонентов.

Владелец почтового ящика может отправлять в него почту со своей машины для дальнейшей отправки во внешний мир с помощью команды RMAIL и принимать из него

прибывшую почту к себе на машину с помощью команды SMAIL. Порядок работы с этими командами аналогичен порядку работы с командами RZ и SZ.

Допустим, ваше письмо содержится в файле greeting.txt (в каталоге отправки вашей телекоммуникационной программы) и вы отправляете его в свой почтовый ящик:

```
F:\> HOME

F:\MAIL\SMIRN> RMAIL

Сервер готов к приему файлов протоколом Zmodem
```

После отправки файла greeting.txt в ваш почтовый ящик он автоматически будет переименован в файл с новым именем и расширением «o» (out — наружу). Через некоторое время его заберет из вашего ящика специальная почтовая программа и отправит дальше по электронной сети передачи данных.

И уже на следующий день, заглянув в свой почтовый ящик на сервере, вы можете обнаружить ответ вашего адресата. Прибывшая почта содержится в файлах с расширениями «i» (in — внутрь):

```
F:\MAIL\SMIRN> DIR

  16876.i   from: rector@univer.omsk.su

F:\MAIL\SMIRN> TYPE 16876.i

Received by tit-bit.msk.su at 17:41:35 +003
Received by kiae.su at 17:02:12 +003
Received by host.omsk.su at 15:48:43 +007
Received by univer.omsk.su at 15:06:34 +007
to: smirn@tit-bit.msk.su
from: rector@univer.omsk.su
subject: благодарность

Здравствуйте. Благодарю за приглашение. Обязательно приеду с визитом в
Ваш институт.
                                С уважением, Сергей Петров
```

На конверте полученного вами письма проставлены отметки и время получения письма всеми машинами компьютерной сети, через которые оно прошло, прежде чем попало к вам (на практике конверт полученного письма обычно значительно больше).

Теперь это письмо можно удалить из почтового ящика или принять на ваш компьютер с помощью команды SMAIL. Эта команда принимает из почтового ящика все файлы с расширением «i» (входящая почта).

```
F:\MAIL\SMIRN> SMAIL

Сервер готов к пересылке файлов протоколом Zmodem

F:\MAIL\SMIRN> STOP

Разъединение. Длительность сеанса (минут): 9
```

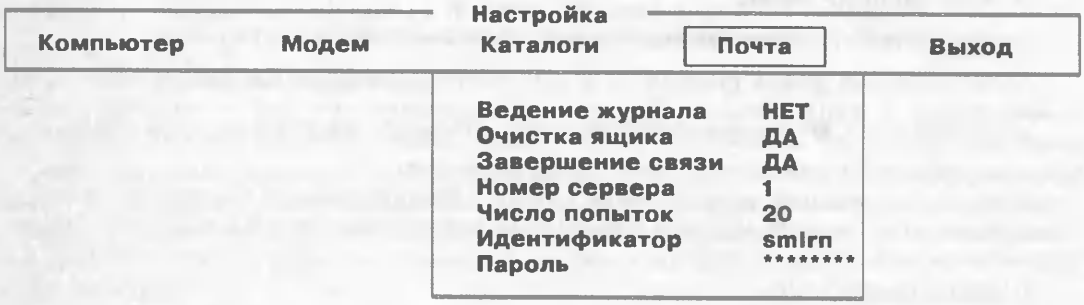
Автоматический прием/отправка электронной почты

Как вы могли заметить, прием/отправка электронной почты является достаточно строго определенной процедурой. Для того чтобы принять и отправить почту, абоненту необходимо дозвониться до телекоммуникационного сервера, ввести свой идентификатор и пароль, перейти в свой почтовый ящик с помощью команды HOME, принять почту с помощью команды SMAIL, отправить почту с помощью команды RMAIL, завершить сеанс связи с помощью команды STOP.

Современные телекоммуникационные программы имеют возможность задать сценарий автоматической работы, который позволяет программе произвести всю необходи-

мую последовательность действий при связи с сервером без вмешательства абонента. Это очень удобно, так как абоненту, которого интересует только работа с электронной почтой, нет необходимости сидеть возле компьютера, ожидать соединения и вводить команды.

ПО TIT-BIT также имеет сценарий автоматической работы, настроить который можно, находясь в режиме «Настройка» и выбрав пункт горизонтального меню «Почта»:



Здесь пункт «Ведение журнала» отвечает за запись протокола автоматической работы в файл tit-bit.log. Пункт «Очистка ящика» определяет, следует ли очищать ваш почтовый ящик на сервере после успешного приема почты на ваш компьютер (фактически этот пункт определяет, какую команду использовать для приема почты: SMAIL или SZ). Пункт «Завершение связи» определяет, следует ли обрывать связь после успешного обмена почтой (давать ли команду STOP). В пункте «Номер сервера» следует указать порядковый номер телекоммуникационного сервера сети TIT-BIT из телефонной книжки, до которого вы будете производить дозвон. Пункт «Число попыток» определяет, сколько раз следует пытаться повторить соединение при установленной некачественной связи. В пункт «Идентификатор» занесите свой идентификатор входа на сервер, в пункт «Пароль» — свой пароль входа на сервер (отображается в виде знаков «*»).

Все, что остается сделать абоненту при автоматическом обмене электронной почтой, — это поместить письма для отправки в каталог отправки \TIT-BIT\SEND, загрузить ПО TIT-BIT в автоматическом режиме работы с помощью запуска файла tit-bit.bat (а не tit-bit.exe), прочитать полученную почту из каталога приема \TIT-BIT\RECEIVE. Внимание: при успешной отправке почты почтовые файлы из каталога \TIT-BIT\SEND удаляются.

Теперь возможности электронной почты могут быть доступны и вам.

Напоминаем телефонные номера телекоммуникационных серверов образовательной сети TIT-BIT:

Москва: (095) 924-7864
Ярославль: (0852) 32-11-83
Архангельск: (8182) 49-06-26
Переславль-Залесский: (08535) 2-05-52

Телефон для консультаций:
(095) 923-0161, E-mail: admin@TIT-BIT.msk.su

Объединение

ИнфоМир

КуМир-Гипертекст

для IBM PC

КуМир-Гипертекст Базовый комплект - обучающая программа по школьному алгоритмическому языку, системе КуМир и исполнителям; практикумы с возможностью получения подсказки по задаче и проверки ответа; возможность проведения автоматизированных контрольных работ; демонстрационные гипертексты для уроков информатики; мгновенная и наглядная диагностика всех ошибок ученика.

Расширенный комплект включает инструментальные средства для подготовки гипертекстов и внешних исполнителей.

На базе системы КуМир-Гипертекст:

Пакет Планиметрия (позволяет производить все геометрические построения и решать задачи на компьютере);

Пакет Функции и Графики (предназначен для написания обучающих программ по математике);

Пакет Информатика-Десять (включает поурочный гипертекст "Основы алгоритмизации" и 6 контрольных работ).

Система КуМир

Полная программная поддержка учебника А.Г.Кушниренко

для Apple Macintosh, УКНЦ, Корвета, Ямахи

Подготовьте Ваш Гипертекст!

Заявки на приобретение программных средств направляйте по адресу:

103051, Москва, Садовая – Сухаревская, д.16, комн.9
"Информатика и образование"

Д. А. Богданова,
научный сотрудник Института проблем информатики РАН, Москва
А. А. Федосеев,
кандидат технических наук, заведующий сектором Института проблем информатики РАН, Москва

ДИСТАНТНОЕ ОБУЧЕНИЕ: МОДЕРАТОР ТЕЛЕКОНФЕРЕНЦИЙ

Статья посвящена новой и актуальной теме – дистантному (дистанционному, удаленному) методу обучения. Основное внимание уделяется возможностям телекоммуникационных систем как средству обеспечения взаимно-обратной связи преподаватель–студент и специфической фигуре модератора – ведущего телеконференции.

Телекоммуникационные системы поддерживают различные режимы связи: доступ к удаленным базам данных, электронную почту и др. В данной статье мы рассмотрим использование такого мощного в смысле образовательных возможностей режима, как телеконференция. Этот режим позволяет организовать и смоделировать такие привычные виды занятий, как семинар, контрольная работа и даже экзамен.

Что же такое телеконференция? Это специальным образом организованная область памяти на компьютере, поддерживающем работу телекоммуникационной системы. Все абоненты, имеющие доступ к этой области памяти (к телеконференции), имеют возможность как получить на свой компьютер весь текст, который уже находился к этому моменту в этой области памяти, так и добавить к нему свой текст. По мере добавления к телеконференции текстов и реплик, присылаемых ее участниками, общий текст становится все более похожим на стенограмму обычной конференции. Отсюда и название — телеконференция.

Существует много видов телеконференций, отличающихся способами взаимодействия ее участников с компьютером (пользовательским интерфейсом), а также способами организации рубрик телеконференции. Различия определяются тем программным обеспечением, которое использует телекоммуникационная система для реализации режима телеконференций.

Однако несмотря на различие телеконференций, всем им присуща одинаковая *структура*. Конференция начинается некоторым текстом, задающим ее тему. Далее каждый из участников имеет возможность добавить к этому тексту свою реплику. Все реплики располагаются последовательно по мере поступления и доступны вместе с исходным текстом всем участникам телеконференции. При последующих обращениях можно получать либо весь текст, либо только новые фрагменты текста. Каждый участник телеконференции имеет возможность работать в удобное для него время.

Участники телеконференции могут быть разбиты на группы для разработки отдельных тем, их доступ к отдельным темам может быть ограничен. Преподаватель может задавать наводящие вопросы, ставить новые проблемы, обращаться к отдельным участникам индивидуально. В общем, телеконференция предоставляет широкие возможности для организации учебного процесса. Однако, каковы бы ни были задания или смысл всей телеконференции, это коллективная деятельность особого рода. Участники этой деятельности не видят друг друга, возможно незнакомы и никогда не познакомятся лично. Их работа в телеконференции растянута во времени и происходит, как правило, на фоне основной деятельности, возможно не имеющей отношения к изучаемому материалу. Как бы то ни было, поведение участников телеконференций оказывается подверженным некоторым закономерностям, зная которые можно эффективно влиять на успешность самой телеконференции и, как следствие, успешность изучения того учеб-

ного материала, усвоению которого телеконференция посвящена.

Успех телеконференций в значительной степени зависит от ведущего — *модератора*. Модератор организует и ведет конференцию, оставаясь с участниками от начала и до конца дискуссии.

Можно выделить несколько основных этапов в работе модератора при проведении конференций:

- обеспечение успешного начала (старта) конференции;
- обучение и помощь на начальной стадии;
- поддержка в разработке и развитии темы дискуссии;
- завершение (закрытие) конференции.

Каждый из этапов требует от ведущего различных умений для того, чтобы цель конференции была достигнута, чтобы участники чувствовали себя комфортно и открыто, высказывая свои идеи. Бесспорно, очень важным является выбор сети, программного обеспечения и техники в сочетании с соответствующей документацией и рабочими материалами.

Кроме того, не следует упускать из виду, что конференции могут подразделяться:

- по способу *доступа* — на «открытые» (каждый, кто использует систему конференций, может присоединиться, если захочет) и «закрытые» (когда только модератор имеет возможность приглашать принять участие в конференции);
- по способу *участия* — на добровольные и обязательные (в конференциях с обязательным участием ведущий может, например, еженедельно менять состав участников конференции, в добровольных принимают участие все желающие обменяться мнениями);
- по способу *достижения цели* — на строго придерживающиеся поставленных для рабочей группы целей или допускающие отступления от основной темы (но в рамках ее).

Рассмотрим более подробно обязанности модератора на каждом этапе проведения конференции.

Успешное начало

Во всех конференциях в своем стартовом послании модератор должен дать ясное обоснование цели конференции. В это послание также должны быть включены расписание и темы для обсуждения. Обычно конференции начинают сразу эффективно работать, если они хорошо ориентированы на определенную тему.

Конференция пройдет успешно, если модератор тщательно отберет участников. Желательно, чтобы общий интерес и общая потребность в обсуждении проблемы явились хорошим стимулом для активного участия в конференции. Модератор должен разослать персональные приглашения потенциально ценным участникам (гостям) конференции в виде «частных приглашений». Кроме того, модератор должен решить, будет конференция проходить без предварительной личной встречи всех участников или такая встреча может оказаться полезной.

Участники должны быть надлежащим образом подготовлены для работы в телеконференции. Это особенно важно для тех участников, кто имеет минимальный навык работы с компьютером или с программным средством телеконференции.

Модератор должен определить продолжительность работы конференции и известить об этом участников, чтобы они могли планировать свое время.

И, наконец, число участников конференции должно быть достаточным, чтобы образовать «критическую массу». Существуют различные мнения относительно того, какое количество участников для этого необходимо, но маленькие группы (10—15 человек) представляются менее эффективными.

Обучение и помощь на начальной стадии

Модератор должен решить, как он сможет определить успех конференции, какие критерии будут взяты за основу (частота посланий, количество новых идей, согласие

между участниками дискуссии). Например, преподаватель, представляя студентам компьютерную конференцию, может быть заинтересован в каких-либо интересных посланиях за определенный промежуток времени.

Так же как хороший ведущий способствует успеху совещания, хороший модератор обеспечивает достижение целей конференции. Модератор, способный организовать дискуссию, успешно проведет конференцию через начальные фазы.

Модератор должен удостовериться, что все участники зарегистрировались в конференции и не испытывают никаких технических сложностей. В то же время он должен быть готов к возможным проблемам, связанным с трудностями работы с клавиатурой, а также с недостаточной грамотностью участников. Людям бывает трудно преодолеть страх написать что-либо неграмотно. Модератор должен сделать очевидным для участников, что очень важным и ценным является обмен идеями и что грамотность в большинстве случаев не столь существенна.

Важным фактором успеха конференции является выбор времени начала ее работы, так как неудачное или позднее начало в значительной степени влияет на желание работать.

Одним из видов деятельности модератора является работа с идеями, которые возникают в ходе конференции. Модератор должен определить, когда наступило надлежащее время поднять новый вопрос для обсуждения. Если участники «перепрыгивают» с одной темы на другую или в рамках объявленной темы обсуждается много разных идей, модератор должен разделить конференцию на несколько дискуссий.

Большое значение имеет способ принятия коллективного решения. Голосование — хорошая, но трудно реализуемая возможность. Другой, более приемлемый, способ — предложить участникам некоторое решение и спросить их мнение, объявив при этом, что если до определенной даты не возникнет каких-либо веских возражений, модератор будет считать предложенное решение решением всей группы.

Поддержка в разработке и развитии темы дискуссии

Очень важно осознать, что модератор конференции должен выполнять разные функции. Он является организатором и должен стремиться к нахождению наилучших решений, обеспечивая успех конференции. В этой стадии его роли могут быть следующими:

- *постановщик целей* — строит планы, но умеет определить точку, в которой изменение этих планов может быть очень существенным;
- *дискриминатор* — дифференцирует полезные и бесполезные идеи, поощряя развитие первых и по возможности отсекая вторые, что существенно, поскольку конференция нуждается в сохранении поставленной цели. Замечено, что как только участники осваивают программное обеспечение телеконференции, возникает тенденция к более многословным и менее содержательным высказываниям, реакция на которые может увести обсуждение от основной темы. Поэтому модератор должен находить способы ограничения таких обсуждений для сохранения основного направления дискуссии;
- *регулятор интенсивности работы конференции* — поддерживает динамику работы группы: возвращает группу к повторному обсуждению при слишком быстром и поверхностном рассмотрении темы; побуждает группу к активности при наметившихся спадах в работе; в случае продолжительного молчания кого-либо из участников выясняет причины и, если необходимо, оказывает помощь;
- *консультант* — анализирует послания участников группы; поднимает заново вопросы, на которые не было найдено ответа; привлекает к обсуждению экспертов, если считает это необходимым;

- *гостеприимный хозяин* — умеет оценить настроение конференции и удостовериться, что участники чувствуют себя комфортно.

Модератор должен вернуть участников конференции к теме обсуждения в случае, если они отошли от нее. Когда становится очевидным, что один или несколько участников конференции уводят ее от поставленной цели, должны быть введены ограничения. Это очень трудная задача для модератора, так как никому не хочется разрушать уровень комфорта в обсуждении. Однако обычно лидирующие члены группы могут нанести большой вред структуре конференции, поэтому модератору надлежит проявить строгость.

Обычно уровень технической подготовленности конференции в значительной степени оказывает влияние на ее развитие. Например, некоторые люди не очень уверенно чувствуют себя, используя новые средства. Поэтому модератор должен, насколько это возможно, гуманизировать технологию.

Если модератор не получает никаких сообщений от участника, то можно предположить, что этот человек не заинтересован в теме конференции. Однако модератор должен проверить, не существует ли каких-либо технических сложностей для этого участника.

Завершение (закрытие) конференции

Наступает момент, когда предмет конференции обсужден со всех сторон. Возможно, хорошей идеей в этом случае будет ограничение времени активной работы конференции, так чтобы участники не чувствовали себя обязанными поддерживать конференцию работающей, когда ее тема уже исчерпана.

Закрывая конференцию, модератор должен поблагодарить участников за их вклад в работу. Может быть полезным за-

ранее напомнить участникам конференции, что она скоро заканчивается. В подобных случаях может наблюдаться всплеск активности в конце конференции.

Обычно закрытие конференции проходит в два этапа. После того как модератор уведомил всех участников о скором закрытии конференции, он может перевести ее в режим «только чтение». Это позволит участникам выделить, какая именно информация им необходима без добавления новых комментариев. Через некоторое время модератор может удалить конференцию из системы.

Общение посредством телекоммуникаций относится к группе сравнительно новых и наиболее быстро развивающихся видов деятельности. Профессия модератора находится в процессе становления и наверняка будет приобретать новые черты по мере развития средств телекоммуникаций и методов их использования.

Литература

Материалы к курсу Открытого Университета (Великобритания) «ONLINE EDUCATION AND TRAINING. How to design and present course using computer networks» (OET Guide, 1992).

В. Н. Пронин,
директор департамента маркетинга и поддержки НЦПСО

MULTIVISION PRO — ГОД 1995

Начавшийся 1995 год принес определенные изменения в маркетинговую политику НЦПСО на рынке образования. Еще не успела высохнуть типографская краска на обложке шестого номера журнала за прошлый год, а мы вновь обращаемся к нашему читателю, стремясь как можно быстрее донести информацию об этих изменениях. Речь пойдет о новых модификациях нашего комплекса — MultiVision Pro.

Сегодняшние перемены в составе комплекса, планируемое появление на рынке его новых модификаций не явилось для нас неожиданностью. Они были подготовлены подробным анализом Российского образовательного и международного компьютерного рынка, исследованиями интересов как потенциальных покупателей так и конкретных пользователей MultiVision Pro, учетом экономического и финансового положения образовательных учреждений, анализом деятельности «конкурентов» (хотя о прямой конкуренции в условиях сегодняшнего рынка говорить, видимо, не приходится).

Непосредственным толчком к предполагаемым изменениям послужила осенняя выставка COMDEX-94, на которой наша фирма была представлена в качестве единственного российского участника своим персональным стендом. На выставке были закреплены старые и установлены новые тесные деловые контакты с нашими зарубежными партнерами.

Основой маркетинговой политики НЦПСО на рынке информатизации Российского образования в этом году станут следующие точки отсчета.

1. Одним из главных направлений нашей деятельности была, есть и останется информатизация предметных областей, внедрение технических и программных средств технологий мультимедиа в преподавание общешкольных (общеучебных) дисциплин. При этом мы предлагаем модификации комплекса, максимально гибко учитывающие как направление, выбранное конкретным учебным коллективом в этой работе, так и финансовые возможности наших клиентов.

2. НЦПСО по-прежнему будет прилагать значительные усилия по технологической, методической и программной поддержке учителей и самого курса информатики, особое внимание уделяя тем авторским коллективам и организациям, кто выбрал для себя нелегкую стезю создания собственных компьютерных и видео-компьютерных учебных курсов и программ, внедрения информационных технологий в реальный учебный процесс.

3. Мы сохраним и дополним «комплексность» предлагаемых нами программно-аппаратных решений как важнейшего фактора формирования модификаций MultiVision Pro, при этом проводя работу по созданию широких возможностей совместимости поставляемого нами оборудования с уже имеющейся в учебных заведениях видео-компьютерной техникой.

4. Организация обучения и постоянного консультирования пользователей комплекса, гарантийные и сервисные обязательства, помощь в получении послегарантийного обслуживания, создание условий для необходимого обновления приобретаемых у нас оборудования и программных средств — эти и другие задачи НЦПСО ставит во главе угла на 1995 и последующие годы.

Основой реализации данной политики станут предлагаемые вашему вниманию модификации комплекса, сохраняющего общее название MultiVision Pro.

В первую очередь речь пойдет о двух более дешевых, чем базовые модели 1994 г., модификациях — MultiVision Pro Value-Priced и MultiVision Pro Value-Priced Desktop. Эти модели ориентированы на пользователей, делающих первые шаги в области информатизации предметных областей, использования видео-компьютер-

MultiVision Pro -

Value-Priced

Компьютер

- IntelliView; Model DPS–3
  486 SLC2 50
  4 Mb RAM
  1.44 FDD
  256 HDD
  8,4" colore Active–Matrix LCD
  Math co–processor
- 14" SVGA monitor 0.28DPI
- KeyBoard

Проекционное оборудование

- Overhead Projector Medium 5000
- SuperLightScreen 115x150
- Stand for Screen 115x150
- Remote Pointer

Мебель

- Рабочий стол учителя Medium PT–7

Программное обеспечение

- Windows 3.11
- DOS 6.22
- CAIRC MultiVision 4.5 (дискеты)

$ 16900

OPTIONS

- Печатающее устройство HP DJ 320
- Digital Option Card
- CD ROM внешний, удвоенной скорости
- Сканирующее устройство HP SJ II cx
- Видеомагнитофон Panasonic
- Видеокамера Panasonic

Value-Priced Desktop

Компьютер

- IBM PC/AT 486
  "MultiVision Pro"
  Intel 486 DX2 66
  8 Mb RAM
  256 Kb CASHE
  450 HDD
  1,2; 1,44 FDD
- 15" SVGA Colore Monitor Tatung 0.28 DPI
- 101 Focus KeyTrack KeyBoard
- MultiMedia Kit Diamand 5000

Проекционное оборудование

- Overhead Projector Medium 5000
- SuperLightScreen 115x150
- Stand for Screen 115x150
- Remote Point
- Online Colore Works SX 8,4" Dual–scan STN панель

Мебель

- Рабочий стол учителя Medium PT–7

Печатающее устройство

- Printer HewlettPackard DeskJet 320

Программное обеспечение

- Windows 3.11
- DOS 6.22
- CAIRC MultiVison 4.5

$ 16999

Basic

Компьютер

- IBM PC/AT 486
  "MultiVision Pro"
  Intel 486 DX2 66
  8 Mb RAM
  256 Kb CASHE
  450 HDD
  1,2; 1,44 FDD
- 15" SVGA Colore Monitor Tatung 0.28 DPI
- 101 Focus KeyTrack KeyBoard
- MultiMedia Kit Diamand 5000

Проекционное оборудование

- Overhead Projector Medium 10K
- Proxima Ovation 820
- Proxima Cyclops 2030
- LazerPointer A90
- SuperLightScreen 115x150
- Stand for Screen 115x150
- Remote Pointer

Мебель

- Рабочий стол учителя Medium PT–7

Видео оборудование

- Видеомагнитофон Panasonic VHS
- Видеокамера Panasonic VHS

Печатающее устройство

- Printer HewlettPackard DeskJet 320

Сканирующее устройство

- Scanner HewlettPackard ScanJet II cx

Программное обеспечение

- CAIRC MultiVison 4.5
- Windows 3.11
- DOS 6.22

$ 23999

По вопросам приобретения и за дополнительной информацией обращайтесь:

НЦПСО

Научный Центр Программных Средств Обучения

Адрес: 109004, г.Москва, ул. Б.Коммунистическая, 9-а
ТЕЛЕФОН. (095) 214 77 84, 214 46 49, Факс (095) 271 04 28

год 1995

Portable

Компьютер

- MultiMedia Notebook
  486 DX4 100
  16 Mb RAM
  450 HDD
  1.44 FDD
  Fax Modem (встроенный)
  CD ROM (встроенный)
  Активная матрица
  640x480
- 101 Focus KeyTrack KeyBoard

Проекционное оборудование

- Proxima DeskTop Projector 2800
- Proxima Cyclops 2060
- LazerPointer A90
- Tripod Screen Professional 150x150

Видео оборудование

- Видеокамера Panasonic VHS

Печатающее устройство

- Printer HewlettPackard DeskJet 320

Программное обеспечение

- CAIRC MultiVison 4.5
- Windows 3.11
- DOS 6.22

$ 34999

Video Work Station

Компьютер

- IBM PC "MultiVision Pro"
  Pentium 90
  32 Mb RAM
  256 Kb CASHE
  1 Gb – IDE4 HDD
  1.2 1.44 FDD
  IDE PCI–controller (4Mb)
  Видеокарта JazzCard:
  Jacarta;
  Projector;
  Port of Entry
- MultiMedia Kit Diamand 5000
- 15" SVGA Colore Monitor Tatung 0.28 DPI
- Big Tower
- MicrosoftKeyBoard
- MicrosoftMouse

Видео оборудование

- Видеокамера Panasonic S–VHS
- Видеомагнитофон Panasonic S–VHS
- Телевизор Panasonic GAO S–VHS

Печатающее устройство

- Printer HewlettPackard DeskJet 1200 C

Мебель

- Рабочий стол учителя Medium PT–7
- Видеоподставка Medium VT–2

Проекционное оборудование

- Overhead Projector Medium 10K
- Proxima Ovation 842
- Proxima Cyclops 2050
- LazerPointer A90
- SuperLightScreen 130x180
- Stand for Screen 130x180
- Remote Pointer

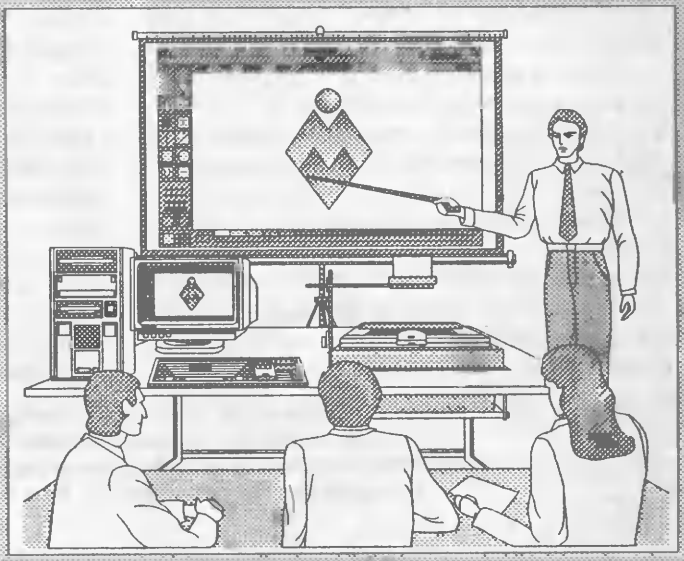
Сканирующее устройство

- Scanner HewlettPackard ScanJet II c

Программное обеспечение

- Windows 3.11
- MS DOS 6.22
- MS Office 4.3
- Adobe Premier
- AIST AutoAnimation 1.0
- AIST MediaMaster 1.2
- CAIRC MultiVision 4.5

$ 45999



Внимание!!!

Обязательства по послегарантийному обслуживанию принимают на себя наши деловые партнеры

А/О "НОВАЯ ЭКСПОЗИЦИОННАЯ ФИРМА"

Москва, ВВЦ, п-н "ФИЗИКА"

тел. (095) 181 96 31

факс (095) 181 23 75

ных средств демонстрации, применения мультимедиальных источников информации, сопровождающих лекцию (рассказ учителя), ответы учащихся.

Оба комплекса компактно размещаются на специализированном рабочем столе учителя, легко перемещаемом в масштабах учебного заведения. Меньшая мощность Overhead Projector Medium 5000 (кодоскопа) с успехом компенсируется поставкой экрана с активным покрытием, увеличивающим яркость свечения красок, а отсутствие ставшего уже привычным для наших пользователей Cyclops (что связано со спецификой представленных моделей проекционных панелей) — наличием Remote Point — радиомыши, осуществляющей дистанционное управление программой как в стандартном учебном классе, так и в лекционном зале.

Изначальное присутствие в MultiVision Pro Value-Priced Desktop более мощного мультимедиального компьютера и цветного принтера делает эту модель (при практически одинаковой цене) более привлекательной для тех, кто планирует для себя полупрофессиональную деятельность по созданию собственных обучающих программ, раздаточных дидактических материалов на уровне поурочных и тому подобных разработок.

В то же время, наличие активной матрицы в компьютере IntelliView DPS-3 создает условия для расширения по желанию пользователя как демонстрационных, так и методических возможностей модели Value-Priced за счет подключения видеоисточников информации и периферийных устройств.

Базовая модель комплекса MultiVision Pro Basic практически не отличается от базовой модификации 1994 г. В нее дополнительно включены радиомышь и экран с активным покрытием SuperLight Screen. Как мы и предполагали, произошла замена печатающего устройства (во всех моделях, где оно представлено) на цветной принтер фирмы Hewlett Parckard — более дешевый и менее критичный по расходным материалам.

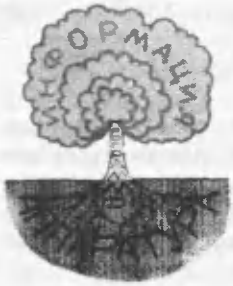
Анонсированная нами в предыдущей статье модификация MultiVision Pro Portable получила свое реальное воплощение на полгода раньше, чем планировалось, при этом мы постарались подобрать комплект входящего в нее оборудования в максимально компактном виде, сделав ее действительно «переносной» моделью с сохранением разработческих мощностей.

Особого внимания заслуживает последняя из представляемых вам моделей — MultiVision Pro Video Work Station. Эта модификация (наиболее «тяжелая» как по комплектующим, так и по ценовым составляющим) включает мощную разработческую машину Pentium 90 Mz, полупрофессиональное видеооборудование формата S-VHS, профессиональное программное обеспечение. Она ориентирована главным образом на пользователей, работающих над реальным созданием собственной видео-компьютерной продукции и тиражированием ее на дискетах, видеокассетах или компакт-дисках. При этом Video Work Station полностью поддерживает все демонстрационные, методические и дидактические возможности комплекса.

Завершая рассказ о новых моделях MultiVision Pro, мы бы хотели обратить еще раз подчеркнуть следующее.

Мы предлагаем комплексное решение проблем по созданию и практической реализации информационных технологий обучения. Но при этом все средства видеокомпьютерной демонстрации, входящие в состав различных модификаций MultiVision Pro, совместимы не только друг с другом (а следовательно и взаимозаменяемы), но и могут быть без значительных технических сложностей подключены к большинству видов уже имеющейся у вас зарубежной видео-компьютерной техники. Это предоставляет широкие возможности выбора модели, подбора комплектности и розничного приобретения этого оборудования.

Более подробную информацию о новых модификациях ПАК MultiVision Pro можно получить в демонстрационном зале департамента маркетинга и поддержки НЦПСО по адресу: Москва, ул. Н. Масловка, д. 16.



ПРОФЕССИОНАЛЬНЫЙ КОМПЬЮТЕР

О. А. Краснова, И. В. Левченко,
преподаватели ОИВТ педагогических колледжей № 13 и 14, Москва

ИЗЛУЧЕНИЕ КОМПЬЮТЕРА И ЗДОРОВЬЕ ДЕТЕЙ

В настоящее время компьютеры все увереннее входят в нашу жизнь. И чем больше становится их вокруг нас, чем больше времени наши дети проводят за ними, обучаясь и играя, тем острее встает вопрос о влиянии излучения компьютеров на здоровье детей и подростков, а значит, на их будущее.

Учителя информатики, безусловно, знают о санитарно-гигиенических нормах длительности занятий учащихся за ПЭВМ [1]. На практике же при проведении компьютерных занятий эти нормы зачастую нарушаются. Нередко на уроках информатики учителя уделяют основное внимание программированию, забывая об общеобразовательных аспектах данного курса. В связи с этим резко возрастает время работы учащихся за компьютерами. На наш взгляд, необходимо не только знать установленный режим работы с вычислительной техникой, но и понимать, чем он обусловлен. А для этого необходимо располагать достаточной информацией по данной проблеме.

В периодической печати регулярно появляются публикации, посвященные влиянию компьютерных занятий на здоровье детей, например, об исследованиях факторов, обуславливающих возникновение нервно-эмоциональной напряженности [2], о выявлении изменения сердечно-сосудистой системы [3], о жалобах на зрительное и общее утомление [4]. Но исчерпывающей информации о последствиях излучения от компьютеров, которыми оснащены учебные учреждения, не поступало. Более того, помимо публикаций о негативных воздействиях компьютерного излучения регулярно можно встретить статьи, в которых отмечается, что «не надо бояться «вреда» от экрана», так как «дети у своего компьютера или телевизора проводят гораздо больше времени» [5], аргументируется непревышение получасового занятия с 3—4-летними детьми (заметьте, 30 минут при норме не более 10 минут для шестилеток!) снижением интереса к компьютерным занятиям [6]. И только в последнее время в печати стали появляться статьи, частично проливающие свет на острую проблему электромагнитной безопасности [7], [8].

На 1-м Международном конгрессе по проблемам экологической безопасности в Стокгольме [9] было отмечено, что излучение персональных компьютеров стимулирует заболевания глаз, изменение кожно-мышечной системы, вызывает стрессовые состояния, кожные заболевания.

До недавнего времени все негативные последствия при работе за вычислительной техникой связывали с высокочастотными электрическими полями. Однако компьютер является также источником широкополосных электромагнитных излучений, ко-

торые включают сверхвысокочастотные и инфранизкочастотные поля. Кроме того, компьютер имеет электростатическое поле. Низкочастотное поле может воздействовать на метаболизм и изменение биохимической реакции в крови на клеточном уровне, в результате чего возможно возникновение симптомов стресса. Это поле нарушает протекание беременности, увеличивает в 2 раза вероятность выкидышей у беременных женщин, а также может способствовать нарушению репродуктивной функции и возникновению рака.

Шведы, собрав наиболее полный банк данных о характеристиках излучения компьютеров, разработали стандарты по электромагнитной безопасности пользователей данной техники, которые являются наиболее жесткими в мире. У нас в стране эти нормы обсуждаются в виде проекта и, возможно, в 1995 г. будут приняты как общегосударственный стандарт.

Каждому учителю необходимо иметь информацию о том,

- к каким последствиям (в том числе и отдаленным) может привести превышение установленного лимита времени работы за компьютером;
- отвечают ли оборудование и класс ВТ, в котором работают он сам и его воспитанники, гигиеническим требованиям;
- каким излучениям подвергается организм;
- каковы основные допустимые нормы у нас в стране и за рубежом;
- каковы мероприятия, позволяющие приблизить характеристики установленной техники к общепризнанному стандарту.

Изучая эти проблемы, педагогический колледж № 14 в сентябре 1994 г. заключил договор с Центром электромагнитной безопасности (ЦЭМБ) на проведение полного обследования электромагнитной обстановки в дисплейном классе, где установлены IBM PC/AT 286. Эти компьютеры были закуплены в 1991 г., и около 130 таких же комплектов используются в учебных учреждениях города. В данном классе установлены 12 ПЭВМ, которые соединены в локальную сеть. В помещении сделано заземление, и на мониторах установлены защитные экраны, закупленные колледжем в 1993 г.

Вначале были выборочно обследованы 4 компьютера, а затем электромагнитная обстановка при всех включенных ЭВМ. Особое внимание было обращено на точку, наиболее вероятного положения пользователя у машины (удаление — 0,5 м).

Результаты обследования показали, что уровень излучения переменного электрического поля на низких частотах (0,02—2 Кгц) превышает норму в 15 раз (норматив — 10 В/м; реально — 150 В/м). Уровень излучения переменного электрического поля на частотах 2—400 Кгц превышает норму в 3,2 раза (норматив — 3 В/м, реально — 7,0 В/м). Уровень напряженности переменного электрического поля в диапазоне частот 10 Кгц—300 Мгц превышает норматив в 2,4 раза. В первой полосе частот (0,02—2 Кгц) значение фонового напряжения сети резко превышает значения полей, излучаемых монитором.

Измерения, сделанные с защитным экраном и без него, показали, что уровень электромагнитного излучения не меняется (должен бы уменьшиться!), а уровень статического поля даже возрастает. При одновременном включении всех мониторов значение полей увеличивается на 5%.

Дальнейшие измерения показали, что заземление не выполняет своей функции и вся электростатика вместо того, чтобы «стекать» в землю, остается на экранах мониторов, отрицательно воздействуя на тех, кто работает за компьютером. Такое положение является следствием того, что работы по заземлению не контролируются соответствующей контрольно-измерительной независимой службой. В результате «земля» вроде бы есть, а фактически ее нет, так как она не выполняет своих функций. Безопасная же зона спереди от экрана монитора находится на расстоянии 1,6 м, слева — 1,9 м, справа — 1,5 м.

Ссылаясь на данные ЦЭМБ, можно отметить, что при обследовании 50 различных моделей мониторов только 15% удовлетворяли положениям шведских стандартов, 54% мониторов имеют такое излучение, что безопасное расстояние достигает 2,5 м.

При таком положении дел, когда пре-

При таком положении дел, когда предельно допустимые значения превышаются в несколько раз, нетрудно сделать вывод, что если мы не примем меры по защите здоровья детей и преподавателей, работающих на компьютерах, то степень риска приобретения перечисленных выше заболеваний резко возрастет.

Подводя итог, хотелось бы отметить, что, поскольку в современном обществе компьютеры становятся неотъемлемой частью повседневной жизни, наша задача — сделать работу с ними не только познавательной и интересной для учащихся, но и безопасной. Основываясь на разработках ЦЭМБ, можно рекомендовать следующее:

1. Проверять правильность выполнения заземления и подводки электропитания от сети.

2. Для экранирования полей использовать специальные защитные фильтры с металлизированным покрытием. Если в классе имеются стеклянные фильтры, то их можно использовать как основу для нанесения специального покрытия, которое не ухудшает видимость, но в то же время полностью защищает пользователя от электромагнитного излучения.

3. Так как корпуса мониторов пластмассовые, можно изнутри покрыть либо все стенки корпуса, либо часть их специальным составом. В другом варианте можно использовать навесные экраны.

4. Приобретая мониторы, целесообразно убедиться в том, что они удовлетворяют шведским стандартам (MPR II).

Вышесказанное совсем не означает, что авторы данной статьи выступают против использования ВТ на занятиях с детьми. Наоборот, будучи преподавателями информатики, мы убеждены в целесообразности внедрения новых информационных технологий и их средств в учебно-воспитательный процесс. Другое дело, что следует грамотно организовывать работу на уроке, использовать различные методы и средства обучения и тем самым свести работу за компьютером к разумной и оправданной норме.

Литература

1. Положение о кабинете вычислительной техники всех типов средних учебных заведений//Информатика и образование. 1990. № 3.

2. *Умрюхин Е., Родионова Т., Просвиркин В. и др.* Индивидуальные психофизиологические особенности школьников при работе на компьютере//Информатика и образование. 1992. № 2.

3. *Федорова М.* Физиологическое обоснование режима обучения школьников при работе на ЭВМ//Информатика и образование. 1994. № 2.

4. *Глушкова Е., Леонова Л., Сазанюк З., Степанова М.* Гигиенические требования к занятиям для дошкольников//Информатика и образование. 1990. № 6.

5. *Пискунова Т. Г.* Я не согласна!//Информатика и образование. 1994. № 2.

6. *Вершинин С В.* Компьютеры для дошкольников//Информатика и образование. 1994. № 3.

7. Электронный смог//Вычислительная техника и ее применение. 1988. № 10.

8. *Юрова Я.* Колебания без выживания//Московский комсомолец. 18.05.1994.

9. *Пальцев Ю., Григорьев Ю., Григорьев О.* Компьютеры порождают экологические проблемы//Финансовые известия. 15.09.1994.

В. А. Файзуллин,
г. Борисоглебск Воронежской обл.

РЕАЛИЗАЦИЯ АЛГОРИТМА ФОРМИРОВАНИЯ МАССИВА С РАЗЛИЧНЫМИ СЛУЧАЙНЫМИ ЭЛЕМЕНТАМИ

Читая в № 3 «Информатика и образование» за 1993 г. статью А. И. Сипливого «Способ получения твердой копии экрана, и не только...», я увидел пример неструктурированной программы, которую можно создать только в языках программирования БЕЙСИК или ФОРТРАН.

Я веду речь о программе, реализующей алгоритм формирования массива с различными элементами. Чего только в этой программе не увидишь: и «холостой» ход в цикле, и «перескоки» из одного цикла в другой при помощи GOTO. Сама задача весьма содержательна, и в ЛОГО я уже реализовал этот алгоритм для одной из игрушек. Но в ЛОГО эта задача вообще элементарна за счет наличия примитива, выдающего истину, если объект является элементом соответствующего списка.

В БЕЙСИКЕ же эта подпрограмма действительно для меня оказалась актуальной при «взвешивании» вопросов и ответов в контролирующих программах, тем более что именно в БЕЙСИКЕ создано наибольшее количество добротных эталонов для контролирующих программ, со временем на текущий ответ, выводом на принтер и простой сменой вопросов и ответов при помощи оператора описания данных DATA.

Все это побудило меня просто механически скопировать предложенную А. И. Сипливым программу. Но при ее отладке возникли сложности: массив с раз-

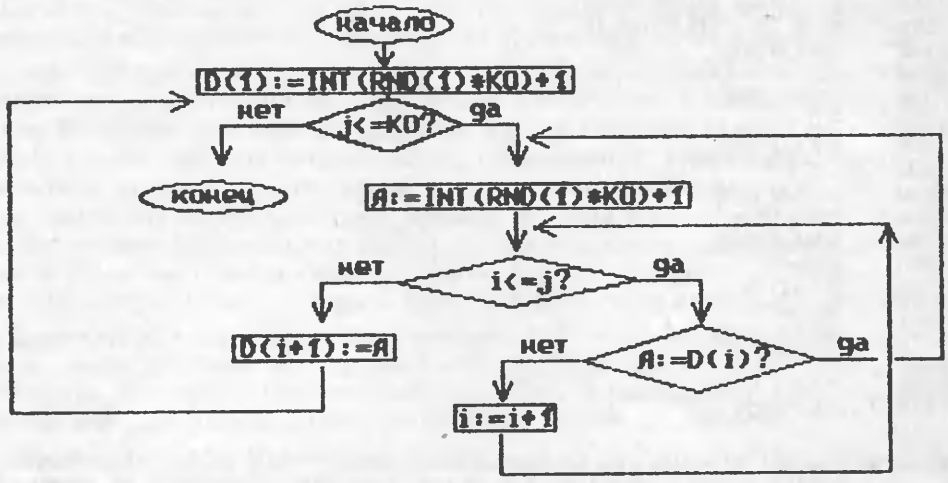


Рис. 1

личными элементами получаться не захотел, разобраться до конца с программой А. И. Сипливого я не смог. Работая с ЛОГО и ПАСКАЛЕМ, я частенько не утруждал себя черчением блок-схем, но сейчас понял, что, работая с БЕЙСИКОМ, без этого не обойтись.

В результате получилась блок-схема, представленная на рис. 1.

Исходя из принципов структурного программирования, я решил, что целесообразно также выделить подпрограмму для формирования массива случайных чисел, а также подпрограмму, обеспечивающую чтение нужных данных «холостым» считыванием данных ненужных. В реализации напечатанной выше программы считываются данные для контролирующей программы из 5 вариантов ответов на вопрос.

В подпрограмме, реализующей описанный выше алгоритм, два цикла «— для» получить оказалось невозможным, так как БЕЙСИК «плохо воспринял» выход из цикла для при помощи GOTO. Поэтому получился один цикл «— для», один «— пока», и один «— до». Дальнейшее просто.

```
2000    'Подпрограмма формирования случайного массива
        с неповторяющимися элементами.
2010    DIM D(K0)
2020    FOR I=1 TO KO'цикл обнуления массива
2030    D (I)=0
2040    NEXT
2050    D (1)=INT(RND(1)*KO+1)' первоначальное присваивание
2060    FOR J=1 TO KO-1'цикл формирования 15 неповторяющихся чисел в массиве
2070    A=INT (RND(1)*KO+1)
2080    I=1
2085    IF I>J THEN 2110'увеличивать номер, сравниваемого элемента массива,
        до сверки со всеми ненулевыми элементами
2090    IF A=D(1) THEN 2070' загадывать новое число, если оно уже встречалось
        в массиве
2100    I=I+1
2105    GOTO 2085
2107    'КЦИКЛ
2110    D(J+1)=A' заполнение массива
2120    NEXT J
2160    RETURN
2170    'Подпрограмма холостого считывания данных
2180    RESTORE
2185    IF D(I)=1 THEN 2320
2190    FOR II=1 TO (D(I)-1)
2200    READ A¤
2210    A¤ =""
2220    READ B¤
2230    B¤ =""
2240    READ C¤
2250    C¤ =""
2260    READ D¤
2270    D¤ =""
2280    READ E¤
2290    E¤ =""
2300    READ X¤
2310    X¤ =""
2311    READ Y¤
2312    Y¤ =""
2315    NEXT II
2320    RETURN
```

Всем желающим получить компьютерную реализацию любых своих контролирующих или обучающих, в стиле программированного обучения, программ обращаться по адресу: 397140, г. Борисоглебск Воронежской области, ул. Аэродромная, дом 14, кв. 60, Файзуллину Владимиру Альбертовичу.

Также интересующимся методикой преподавания информатики в I–II классах могу предложить ряд интересных программ, выполненных на ЛОГО, и методические разработки к этим урокам.

КЛУБ «КОРВЕТ»

Л. И. Яромко,
учитель средней школы № 129, г. Минск

ПАРТИТУРА В АЛГОРИТМИЧЕСКИХ РАМКАХ

Автор делится своим опытом работы в музыкальном редакторе MUSIC.COM (разработчики А. Андреев и С. Ларионов, Москва) или, как они еще его называют – музыкальная система МС-89. В качестве примера взяты музыкальные партитуры, расписанные для одновременного звучания сразу двух ПК 8010 Корветов. Автор считает, что MUSIC.COM является настоящим кладом для внеклассной работы учителя-«корветчика», особенно для школ с музыкальным уклоном, хотя, конечно, годится и для общеобразовательных школ.

Когда я впервые «озвучил» свою первую партитуру (трехголосную), то сразу обратил внимание на довольно тихое звучание мелодии и тогда же попробовал одну и ту же партитуру загрузить на два соседних РМУ и после набора P (PLAY) дать одновременный ввод. Звучание было явно лучшим. Ну а затем пришла мысль: «А почему, собственно, надо засылать одну и ту же партитуру, когда можно расписать ее на две партии и получать, таким образом, одновременное звучание шести разных голосов?» И громче, и красивее. И с тех пор я непрерывно занимаюсь этим редактором.

Но есть у меня к авторам и несколько критических замечаний. И первое из них относится к документации MUSIC.DOC, прилагаемой к музыкальному редактору. Особенно к тому месту, где авторы показывают, как осуществляется кодирование нот в МС-89. Судите сами. Нотоносец у них состоит из 15 линий вместо традиционных пяти. А нот на нем нет и вовсе. И поэтому о кодировке нот в МС-89 можно только догадываться. Не сомневаюсь, что человек, не имеющий специального музыкального образования, в этот музыкальный редактор «не влезет» ни за что на свете по их документации.

Конечно, авторы мне могут возразить — в конце документации говорится, что можно ознакомиться с партитурами HANDEL.MUS и CHORUS.MUS и самому создать нечто подобное. Возможно, но тогда авторам надо было обязательно указать, в каких нотных сборниках можно отыскать вышеуказанные произведения, и дать возможность сравнивать нотную и машинную партитуры. И, кроме того, произведения эти довольно малоизвестные широкому кругу любителей музыки. На мой взгляд, надо было в качестве демонстрационного примера взять всем хорошо известную мелодию (например, «Подмосковные вечера» или хотя бы «Чижик-пыжик») и на их примере показать кодирование в МС-89. Ведь самое главное на первом этапе — видеть сразу и ноты, и их кодировку.

Я, например, вначале показал учащимся рис. 1, 2, 3 (см. далее), затем пару примеров и к концу первого урока они озвучили песню «Перепелочка». А на втором мы уже занялись двух- и трехголосными партитурами. Однако опять вернусь к замечаниям. На мой взгляд, авторы в документации допустили несколько неточностей:

- символ % они почему-то называют «ключ до», хотя это самый настоящий знак бекар (♮);
- символы «'» или «"», по мнению авторов, вызывают режим исполнения стаккато, но точнее здесь было бы применить термин «non legato», т. е. «исполнять не очень

связанно»; а вот символ «,» (запятая), записанный после ноты, как раз и вызывает режим стаккато;

- к счастью, авторы ошиблись, утверждая, что знак «басовый ключ» «может применяться в пределах данного голоса только один раз». Нет. Практически — сколько надо, столько раз и можно применять. А к счастью, потому, что весь аккомпанемент мелодий как раз и держится на чередовании басовых и скрипичных нот (см. далее пример аккомпанемента).

И еще. Авторы «забыли» указать, что действие знака ♯ или ♭ распространяется на «помеченную» ноту до конца такта.

Однако главные трудности освоения позади, и я от работы с МС-89 получаю в основном удовлетворение.

Считаю, что любой желающий может достать ноты в местной нотно-музыкальной библиотеке и, глядя на ноты и на «машинную» программу, освоить программирование в МС-89. В качестве примера «стерео»-партитуры лучше всего подходит русская народная песня «По муромской дорожке» (см. листинг в конце статьи). Она и по объему не самая большая (всего 34 такта), и по звучанию самая удачная.

Введение в МС-89

Обозначения нот в скрипичном и басовом ключах, а также их длительностей приведены на рис. 1, 2, 3.

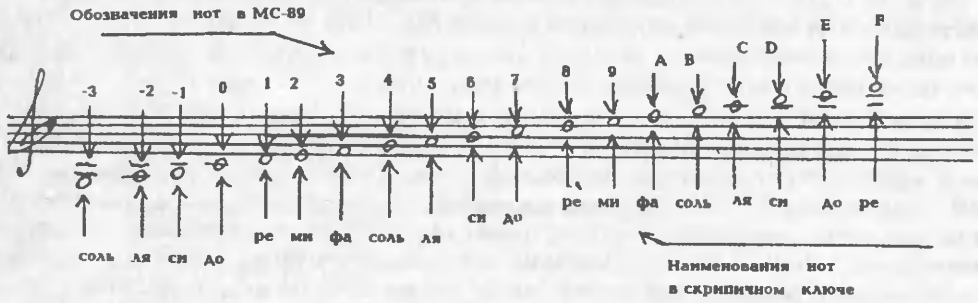


Рис. 1

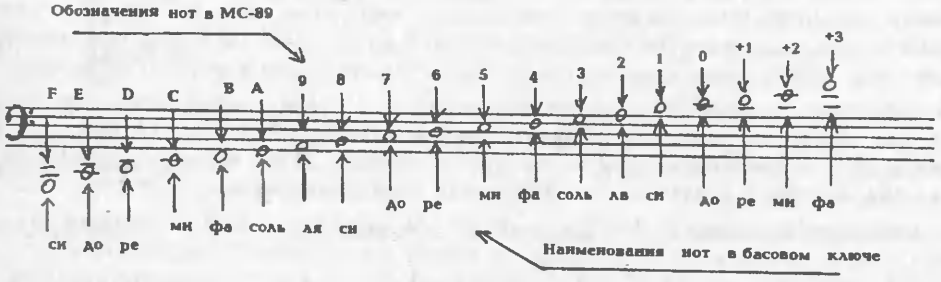


Рис. 2

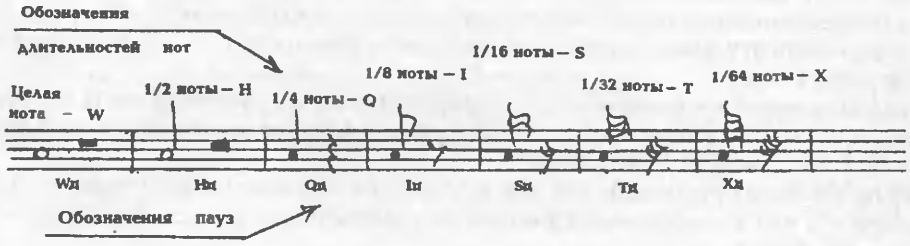


Рис. 3

Скрипичный ключ обозначается символом *. Басовый ключ обозначается символом @. M1, M2, M3, ... — обозначения тактов. V1, V2, V3 — обозначения голосов (в MC-89 их всего три).

Примеры кодирования нот:

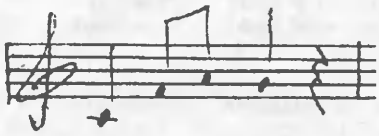


Рис. 4

Кодировка: *Q0I3I5Q4Q♩ или *Q0I35Q4♩.
Примечание: знаки длительности перед каждой нотой можно опускать, если они чередуются

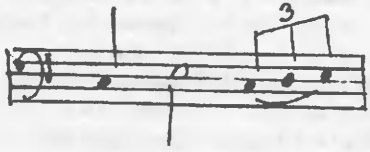


Рис. 5

Кодировка: @Q7H5I:765 (I:765 – кодировка триоли)

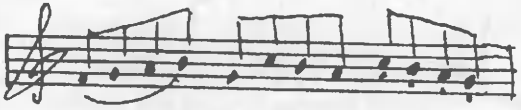


Рис. 6

***I34564'7'6'5'7,6,5,4,**
Пояснение: первые четыре ноты записаны в режиме legato, следующие четыре – в режиме non legato, и последняя четверка – в режиме staccato.

Более качественный (или «более выраженный») режим non legato или staccato можно получить следующим образом:

а) для получения non legato можно «забирать» у каждой ноты 1/4 часть ее длительности и заменять эту часть паузой. Так, например, вместо ✻ Q 1" 2" 4" можно набрать ✻ I.1S¤I.2S¤I.4S¤;

б) для получения же режима staccato можно «забирать» уже 1/2 часть ее длительности и так же поступать со второй ее половиной, да еще и вводить символ «,». Так, вместо ✻ Q 3, 5, 7, получим: ✻ I3,¤5,¤7,¤.

Это необходимо учитывать, так как в MUSIC.COM режим исполнения, вызываемый символами «"» или «'», практически ничем не отличается от режима legato, да и staccato далеко не идеальное.

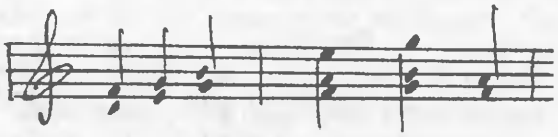


Рис. 7

Кодировка: 10 M1 V1 ✻ Q346 V2 ✻ Q124
20 M2 V1 ✻ Q9B¤ V2 ✻ Q565 V3 ✻ Q343

Осталось познакомиться со знаками, повышающими или понижающими ноты на полтона.

Знак диез (♯) — обозначается символом #;
знак бемоль (♭) — обозначается символом &;
знак бекар (♮) — обозначается символом %.

Ключ альтерации, т. е. количество бемолей или диезов, соответствующих данной тональности, обозначается символом K. После него указывается количество диезов или бемолей, постоянных для данного произведения. Он применяется в партитуре только один раз.

И еще. В MC-89 действует следующий закон: знак #, & или %, поставленный после ноты, действует для этой ноты до конца такта, как в пределах данного голоса, так и в остальных (если нота находилась в первом голосе). Собственно, так же обстоит дело и в нотной грамоте.

Примеры:

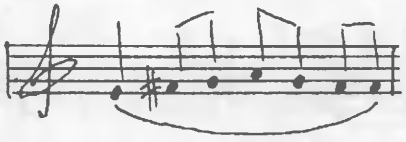


Рис. 8

Кодировка ✻Q2I3#45433
(последние две ноты будут исполняться как фа ♯)

Пример начала музыкального произведения, имеющего в ключе альтерации три бемоля:

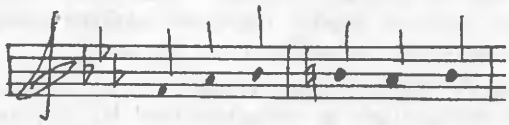


Рис. 9

```
Кодировка:  10  K3& M1V1 * Q356
            20  M2 V2 * Q6%56
```

Пример кодирования такта сопровождения мелодии (аккомпанемента):

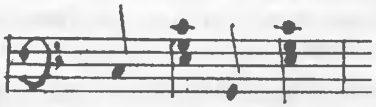


Рис. 10

```
V1@Q7, *0@A, *0, V2@Q7, *2, @A, *2, V3@Q7, *4, @A, *4,
Пояснение: аккорд ми-соль-до переведен из басового ключа в скри-
пичный для более приятного звучания.
```

Редактирование

С помощью команды LIST (L␣n1 или L␣n1 n2) можно вызвать строку (или строки) и, нажав клавишу «забой» (⇦), поднять курсор на любую из них.

Двигаться по строке слева направо можно с помощью комбинации УПР+X. «Уйти» с любой строки можно с помощью клавиши <ВВОД>. К сожалению, в МС-89 отсутствует режим вставки, что, конечно, затрудняет редактирование. Удаление символов производится клавишей <ПРОБЕЛ>.

Просмотр всей программы осуществляется с помощью L, <ВВОД>. Остановка просмотра программы осуществляется нажатием любой черной клавиши. При остановке просмотра с помощью клавиши F3 (или F1) можно вызвать на экран по две следующие строчки. Комбинация <УПР>+L сразу же переводит курсор в левый верхний угол экрана. Спуск — с помощью <ВВОД> и УПР+X.

Запуск (озвучивание) программы

Надо иметь в виду, что набранная программа не будет звучать, если в начало программы не вставить символы P и N. Что же они обозначают и для чего нужны?

С помощью символа P и следующей за ним любой буквой латинского алфавита в МС-89 обозначается начало любого фрагмента (участка) данного музыкального произведения, а если фрагмент нуждается в повторении, то выделяются его начало и конец.

В любом случае надо зафиксировать начало музыкального произведения (например: PA).

Рассмотрим следующий пример. Пусть некоторое музыкальное произведение состоит из 30 тактов. Если вы наберете в программе перед первым тактом сочетание PB, а перед

15-м тактом — PA, то машина исполнит вначале такты с 15-го по 30-й, а затем только с 1-го по 14-й. Если вы поменяете местами PB и PA, то произведение начнет исполняться «нормально», т. е. с 1-го по 30-й по порядку. Впрочем, во втором случае PB можно убрать и вовсе — результат будет тем же. Таким образом, напрашивается вывод: машина исполняет помеченные участки строго в алфавитном порядке.

Ну а с помощью символа N задается темп исполнения музыкального произведения. Надо только после него записать одну из четырех букв H, Q, I или S и после знака равенства указать какое-нибудь число (оно характеризует циклы задержки). Например: NS=75. Буква и число для каждого произведения подбираются экспериментально.

Итак, если вы ввели в начало вашей программы символы P и N, то для прослушивания вам останется только набрать PLAY и нажать <ВВОД>.

Организация повторов в МС-89

Чтобы организовать повторение какого-либо фрагмента музыкального произведения, применяется символ R. После него надо указать букву (метку), стоящую в начале участка, нуждающегося в повторении.

Рассмотрим организацию повторов на следующих примерах.

Пример 1. Возьмем опять 30-тактное произведение и допустим, что надо повторить такты с 15-го по 30-й. Распределим метки следующим образом: перед первым тактом наберем PA, перед пятнадцатым — PB, после тридцатого — PC␣RB — и получим повторение тактов с 15-го по 30-й.

Пример 2. Пусть в этом же 30-тактном произведении нотные знаки повтора :‖ стоят в начале 15-го и в конце 28-го тактов, а знаки 1. , 2. сигнализируют о том, что при повторе надо исполнить такты с 15-го по 26-й включительно, а затем, минуя 27-й и 28-й такты, «перепрыгнуть» сразу на 29-й. Поступим на этот раз следующим образом: перед первым тактом наберем PA, перед пятнадцатым — PB, перед двадцать седьмым — PC, перед двадцать девятым — PD␣RB, а затем (можно в следующей строке) PE.

И поставленная задача будет решена. Вероятно, этих двух примеров вполне достаточно, чтобы в будущем организовать любой повтор в ваших будущих партитурах.

Рекомендации учителю

Записать MUSIC.COM на диск с СОС «Альтернатива» и использовать ее при загрузке. Загружается музыкальный редактор довольно быстро — около 3 мин на 12 РМУ.

После загрузки с помощью команды HELP вызывается список всех команд, имеющихся в арсенале МС-89, и потом достаточно набирать только первую букву любой из них, чтобы добиться их исполнения в среде MUSIC.COM.

Учащиеся начинают набирать программы с места нахождения курсора (под заставкой *МС-89*), вводя каждую строку с помощью клавиши <ВВОД>. Желательно (на начальном этапе) каждую строку после ввода компилировать (C, <ВВОД>) и получать сообщения о наличии возможных ошибок. Прослушивание мелодии можно в любой момент прервать нажатием клавиши <ПРОБЕЛ>. Запись набранных на РМУ партитур на «головной» диск осуществляется с помощью W␣«имя файла» (пробел после W обязателен). Расширение MUS присваивается автоматически.

Загрузку, просмотр и прослушивание созданных на РМУ программ можно осуществлять следующим образом:

I вариант

1. A>MUSIC
2. R␣«имя файла» (без расширения), <ВК>
3. L (или STOP), <ВК>
4. P, <ВК>

II вариант

1. A>MUSIC
2. P␣«имя файла», <ВК>
3. L, <ВК>

Если вдруг при записи на диск произойдет «зависание» ученической и головной машины, можно поступить следующим образом:

- отключить провод «локальная сеть» на «зависшем» РМУ;
- сделать сброс на головной машине;
- снова загрузить СОС «Альтернатива»;
- на любой действующей ученической машине набрать простенькую программу (если нет готовой) типа

```
10 PA
20 NS=100
30 M1 V1*Q123
```

и дать W␣«имя файла». (Например, W␣"PROBA) и дождаться окончания записи;

- снова подключить провод ЛС на «зависшем» РМУ, и созданная программа начнет «сбрасываться» на диск.

Для надежности можно тут же еще раз записать созданную партитуру под каким-нибудь другим новым именем и для проверки сразу же вызвать ее на свободное РМУ и прослушать (или просмотреть). Вообще, это лучше делать постоянно, так как случается, что при записи файла на диск внешне все выглядит благополучно, но в дальнейшем, при попытке его загрузить вдруг появляется сообщение «Ошибка синтаксиса партитуры», и мы понимаем, что время на создание программы потеряно зря. Поэтому лучше (пока программа находится в ОЗУ) потратить одну-две минуты на проверку, чем иметь возможность потерять намного больше времени и пережить в дальнейшем чувство досады.

И последнее. В случае возникновения непредвиденной ситуации при работе с программой (случайное нажатие другой клавиши, появление какого-то сообщения, отсутствие звучания и т. д.) можно всегда выйти из нее, нажав клавишу <STOP>, а затем (к примеру) L, <ВВОД> и снова вернуться к программе.

В заключение приводим одну из практических работ автора. —программу для известной народной песни «По муромской дорожке».

I партия

```
 10 PA
 20 NS=70
 25 K3&
 30 M1 V1*QB V2*Q9
 40 M2 V1*HCQA V2*HAQ8
 45 V3@QB,*0,0,
 50 M3 V1*H8QC V2*H7QA
 55 V3@QB,*0,0,
 60 M4 V1*HBQ9 V2*H9Q7
 65 V3@QA,*0,0,
 70 M5 V1*H7Q4 V2*H4Q4
 75 V3@QC,*0,0,
 80 M6 V1*H6Q5 V2*H.3
 85 V3@QB,*0,0,
 90 M7 V1*H4Q3 V2*H2Q3
 95 V3@QA,*3,3,
100 M8 V1*H.0
105 V2@Q7,*0,0, V3@7,*4,4,
110 M9 V1*Q0$-3
115 V2*Q0,$$ V3*Q4,$$
116 PB
120 M10 V1*H0Q0
125 V2@Q7,*0,0, V3@Q7,*4,4,
130 M11 V1*H1Q1
135 V2@QA,*4,4, V3@QA,*6%,6,
140 M12 V1*H.2
145 V2@Q7,*0,0, V3@Q7,*4,4,
150 M13 V1*H0Q0
155 V2@QC,*0,0, V3@Q7,*4,4,
160 M14 V1*H7Q6%
165 V2@QB,*0,0, V3@QB,*5,5,
170 M15 V1*H7Q5
175 V2@Q8,*1,1, V3@Q8,*5,5,
180 M16 V1*H.4
185 V2@QC,*4,4, V3@QC,*6,6,
190 M17 V1*Q4$6 V2*Q2%$4
195 V3@Q7,*0,0,
200 M18 V1*H5Q3 V2*H3Q1
205 V3@QB,*0,0,
210 M19 V1*H1Q5 V2*H1Q3
215 V3@QB,*0,0,
220 M20 V1*H4Q2 V2*H2Q0
225 V3@QA,*0,0,
230 M21 V1*H0Q4 V2*H0Q2
235 V3@Q7,*0,0,
240 M22 V1*H6Q5 V2*H4Q3
245 V3@QB,*0,0,
250 M23 V1*H4Q3 V2*H2Q1
255 V3@QA,*4,4,
260 M24 V1*H.7 V2*I012345
265 V3@Q9,*0,0,
270 M25 V1*H7Q7 V2*I6789%AB
275 V3@QA,*0,0,
280 M26 V1*H3Q1
285 V2@QB,*3,3, V3@QB,*5,5
290 M27 V1*H1Q3
295 V2@QB,*3,3, V3@QB,*5,5
300 M28 V1*H.2
305 V2@QA,*2,2, V3@QA,*4,4,
310 M29 V1*H0Q2
315 V2@QC,*2,2, V3@QC,*4,4,
320 M30 V1*H4Q3
325 V2@QB,*3,3, V3@QB,*5,5,
330 M31 V1*H2Q3
335 V2@QA,*3,3 V3@QA,*6%,6,
340 M32 V1*Q0I2479
345 V2@Q7,*2,2, V3@Q7,*4,4,
346 PC
350 M33 V1*I876543
355 V2@QB,*3,@A, V3@QB,*5,@A,
356 PD
357 PE RB
358 PF RC
359 PG RB
360 PH RC
361 PI RB
362 PK RC
363 PP RB
369 PX
370 M34 V1*Q7$$ V2*0$$
```

II партия

```
 10 PA
 20 NS=70
 25 K3&
 30 M1 V1*Q9 V2*Q7 V3*Q$
 40 M2 V1*H7Q7
 45 V2@QB,*3,3, V3@QB,*5,5,
 50 M3 V1*H5Q8
 55 V2@QB,*3,3, V3@QB,*5,5,
 60 M4 V1*H7Q4
 65 V2@QA,*2,2, V3@QA,*4,4,
 70 M5 V1*H2Q4
 75 V2@QC,*2,2, V3@QC,*4,4,
 80 M6 V1*H.0
 85 V2@QB,*3,3, V3@QB,*5,5,
 90 M7 V1*H-1%Q1
 95 V2@QA,*4,4, V3@QA,*6%,6,
100 M8 V1*H.0
105 V2@Q7,*0,0, V3@Q7,*2,2,
110 M9 V1*Q0$-3
115 V2*Q0,$$ V3*Q2,$$
116 PB
120 M10 V1*Q0I2479
125 V2@Q7,*2,2, V3@Q7,*4,4,
130 M11 V1*Q1I346%8
135 V2@QA,*3,3, V3@QA,*4,4,
140 M12 V1*Q2I46%79
145 V2@Q7,*2,2, V3@Q7,*4,4,
150 M13 V1*Q0I2479
155 V2@QC,*2,2, V3@QC,*4,4,
```

```
160 M14 V1*Q0I8ACD%
165 V2@QB,*0,0, V3@QB,*3,3,
170 M15 V1*Q1I3578
175 V2@Q8,*1,1, V3@Q8,*6,6,
180 M16 V1*Q2I6543
185 V2@QC,*2,2, V3@QC,*4,4,
190 M17 V1*Q2%I7646
195 V2@Q7,*2%,2, V3@Q7,*6,6,
200 M18 V1*Q3I578A
205 V2@QB,*3,3, V3@QB,*5,5,
210 M19 V1*Q1I3578
215 V2@QB,*3,3, V3@QB,*5,5,
220 M20 V1*Q2I4798
225 V2@QA,*2,2, V3@QA,*4,4,
230 M21 V1*I765420
235 V2@Q7,*2,2, V3@Q7,*4,4,
240 M22 V1*H1Q0
245 V2@QB,*3,3, V3@QB,*5,5,
250 M23 V1*H.-1%
255 V2@QA,*3,3, V3@QA,*6%,6,
260 M24 V1*I012345
265 V2@Q9,*2,2, V3@Q9,*5,5,
270 M25 V1*I6789%AB
275 V2@QA,*2%,2, V3@QA,*6,6,
280 M26 V1*H7Q7 V2*H5Q5
285 V3@QB,*0,0,
290 M27 V1*H5Q7 V2*H3Q5
295 V3@QB,*0,0,
300 M28 V1*H.7 V2*H.4
305 V3@QA,*0,0,
310 M29 V1*H4Q7 V2*H2Q4
315 V3@QC,*0,0,
320 M30 V1*H7Q8 V2*H6Q5
325 V3@QB,*0,0,
330 M31 V1*H9Q8 V2*H4Q4
335 V3@QA,*4,4,
340 M32 V1*H.7
345 V2@Q7,*0,0, V3@Q7,*4,4,
346 PC
350 M33 V1*Q7$-3
355 V2@Q7,*0,0, V3@Q*4,4,
356 PD
357 PE RB
358 PF RC
359 PG RB
360 PH RC
361 PI RB
362 PK RC
363 PP RB
369 PX
370 M34 V1*Q2&$$ V3*Q4$$
```

ЛИнТех

Хватит мечтать - давайте действовать!
Превратите КУВТ УКНЦ,"Корвет" и БК в IBM-PC

Принципиально новые системы **"NET-Rt11 & DOS-LINE"** и **"NET-CP/M & DOS-LINE"** позволят Вам превратить КУВТ УКНЦ, "Корвет" и БК в классы IBM-PC. На каждом рабочем месте Вы будете работать, как на IBM-PC, под управлением MS-DOS, использовать Norton Commander, Лексикон, Turbo Basic и другие популярные программы для IBM-PC. При этом полностью сохраняется возможность использования всего существующего программного обеспечения для этих КУВТ.

Для модернизации КУВТ достаточно приобрести нашу систему и установить в КУВТ IBM-совместимый головной компьютер.

Локальные сети **" NET-Rt11 & DOS-LINE"** и **"NET-CP/M & DOS-LINE"** объединяют с помощью высокоскоростных сетевых адаптеров в единое целое головной компьютер IBM-PC и ученические машины. Скорость работы повышается в 30-100 раз, на каждом компьютере ученика обеспечивается полноценная работа без сбоев и зависаний благодаря отказу от использования стандартного сетевого оборудования и дисководов.

Цена систем ниже цены одного IBM-совместимого компьютера. В настоящий момент ими оснащено более 300 компьютерных классов на территории России, Белоруссии, Украины и Казахстана.

Все системы просты в установке и использовании, не требуют перемонтажа существующих линий связи, весь процесс модернизации стандартного класса занимает 2-3 часа. Гарантия - 3 года со дня приобретения.

Министерство образования РФ рекомендует
использовать системы
"NET-Rt11 & DOS-LINE" и "NET-CP/M & DOS-LINE"
для модернизации КУВТ "Корвет" и УКНЦ.

ЛИнТех
Телефон/факс: (095) 273-50-14
E-mail: shop@lintech.msk.su
119501, Москва, а/я 942.

MultiVision Version 4.5

ВЫ МОЖЕТЕ СОЗДАТЬ

- Образовательные программы для демонстрационной поддержки учителя и самостоятельного изучения материала учащимися.
- Интерактивные шоу для докладов и лекций.
- Демонстрационные ролики для выставок.
- Путеводители, компьютерные справочники, каталоги и многое другое.

ВОЗМОЖНОСТИ

В своих обучающих или демонстрационных программах Вы можете использовать:

- мультфильмы, для создания которых не требуется профессиональных навыков;
- картинки, нарисованные или считанные с помощью сканера;
- текст с разнообразным шрифтовым оформлением;
- звук, воспроизводимый с высоким качеством.

ПРОСТОТА В ИСПОЛЬЗОВАНИИ

Для создания собственных программ совершенно не обязательно быть программистом. MultiVision позволяет, используя одну только мышь, шаг за шагом воплощать Ваши замыслы, пока Вы не добьетесь желаемого результата.

МОДУЛИ

Всю необходимую информацию для Ваших программ можно подготовить в специальных модулях MultiVision редакторах.

- Animator позволяет быстро создавать мульфильмы. Вы только задаете ключевые кадры, их преобразования и траектории движения.
- Imager собирает изображения и мультфильмы в единые библиотеки.
- PhotoMaster - инструмент для сканирования изображений.
- Painter - мощный графический редактор с уникальным набором возможностей.
- Writer - специализированный редактор для подготовки красочных экранных текстов. Импорт текстов из MS Word 5.0, WinWord 2.0, Word Perfect 5.1. Поддерживает шрифты в формате TrueType.
- Designer - это основной модуль MultiVision, в котором отдельные объекты компануются в законченную программу. Здесь определяется порядок просмотра информации и ее расположение на экране.

РЕЗУЛЬТАТ

Результатом работы MultiVision является не набор файлов, а единая самостоятельная программа, работающая под DOS или MS Windows. Такая программа может быть легко перенесена на любой другой компьютер и распространяться без всяких ограничений.
При желании Вы можете записать программу в виде проигрываемого ролика.

НЦПСО
Научный Центр Программных Средств Обучения
Адрес: 109004,г.Москва,ул. Б.Коммунистическая,9-а
ТЕЛЕФОН: (095) 272 11 25,214 46 49. Факс (095) 271 04 28

MultiVision

"Замечательное средство для создания презентаций
с использованием красочных анимаций, текстов и музыки"
"PC WORLD"

Универсальная среда для
гибкой генерации и сборки учебных
курсов, обучающих программ и
компьютерных шоу.