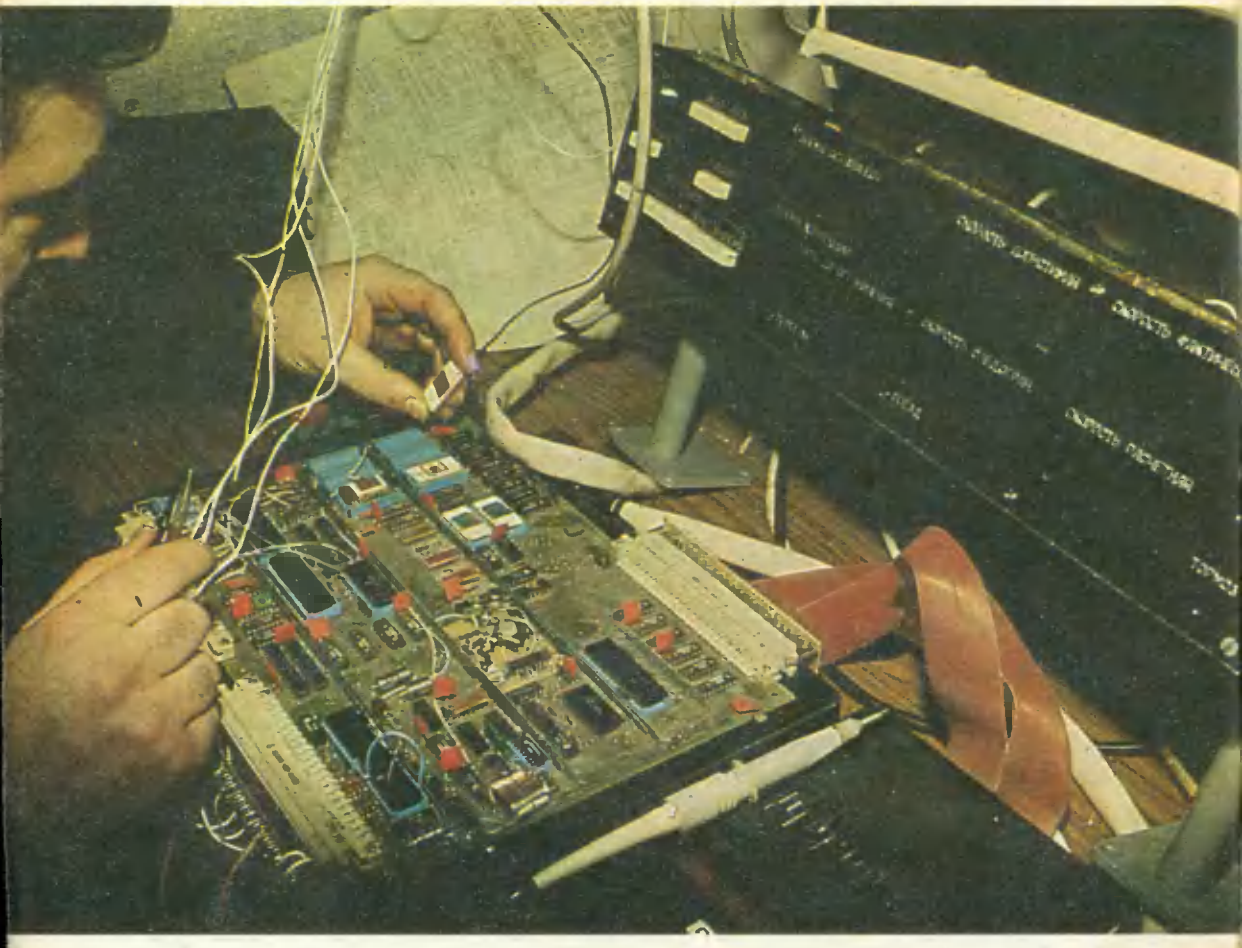
ISSN 0234—0453

ИНФО

ИНФОРМАТИКА И ОБРАЗОВАНИЕ

3 1987



Электроника — это точность и экономичность

Какое отношение к информатике имеет мощный электровоз Р-200, способный везти пассажирский поезд со скоростью 200 км/ч? Ответ прост: платы с микросхемами на четвертой странице обложки — функциональные узлы макетного образца системы автоматического ведения поездов, основанной на микропроцессорах. На других снимках — макетирование системы автоведения (вторая страница обложки) и отладочный комплекс для нее (третья страница). А как она работает и что способна дать железнодорожному транспорту, вы сможете прочесть в статье создателей «электронного машиниста», помещенной в рубрике «ЭВМ в народном хозяйстве».

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

Выходит шесть раз в год
Издается с 1986 г.

Издательство «Педагогика»
Москва

ИНФОРМАТИКА И ОБРАЗОВАНИЕ

3 '87
май — июнь

Содержание

Обложка — *Э. Бажилина.*

В оформлении номера принимали участие — *Э. Бажилин, Р. Бумагин, Н. Истомин, Б. Кувшинов, С. Кустарева, А. Пономарев, С. Расторгуев, В. Шкарбан.*

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

Почтовый адрес: 107847, Москва, Лефортовский пер., 8

Сдано в набор 26.03.87. Подписано в печать 06.05.87. А 07308. Формат 70×100/16. Печать офсетная. Усл. печ. л. 10,40. Уч.-изд. л. 11,49. Усл. кр.-отт. 42,88. Тираж 91 210 экз. Заказ 897. Цена 60 коп.

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

ЕРШОВ А.
Академик

Г. А. Звенигородский

В будущей истории становления школьной информатики в СССР работы Г. А. Звенигородского (1952—1984) по праву займут заметное место. Никакой обзор работ по таким важным разделам, как изучение основ информатики в раннем подростковом возрасте, получение в средней школе профессии программиста, разработка и внедрение учебных языков программирования, без учета исследований и разработок Г. А. Звенигородского не будет полон. Важной особенностью его подхода является органическое объединение тематических и педагогических аспектов работы с учащимися, живой практики и теоретического осмысления материала*.

Геннадий Анатольевич Звенигородский родился 9 августа 1952 г. в семье служащих. Первый опыт работы с учащимися он получил, как и все активные комсомольцы, еще в школе. В его интересах преобладала физика и новая техника, и в 1969 г. он поступил в Харьковский политехнический институт на инженерно-физический факультет.

Учился Геннадий успешно и уже к окончанию в 1975 г. вузовского курса по специальности «Динамика полета и управление» имел несколько публикаций [2, 11] и производственных отчетов.

Важную роль в студенческие годы Геннадия сыграла его общественная работа в межвузовском консультативном пункте для абитуриентов при Харьковском городском комитете ЛКСМУ. Цель этого пункта была ориентировать старшеклассников в специальностях, которые можно было получить в харьковских вузах. Это работа, требовавшая и широкого кругозора, и чувства нового, лекторских и литературных способностей. В течение полугода Звенигородский вел рубрику «Абитуриент-75» в комсомольской газете «Ленинская смена» [3—6]. В этих публикациях виден крепнущий интерес молодого автора к вычислительной технике и программированию, к тому, чтобы делать новые знания достоянием школьников.

Со свойственной ему систематичностью Геннадий ознакомился с опытом работы со школьниками по кибернетике в Малой академии наук в Симферополе, в заочной математической школе при МГУ и других местах. Начав в 1974 г. с руководства отдельными школьными кружками по кибернетике и программированию, он организовал в 1975 г. при Харьковском городском Дворце пионеров и школьников имени П. П. Постышева школу юных кибернетиков [9]. Ее актив составили победители проведенной Звенигородским олимпиады юных кибернетиков [7, 8].

С 1975 г. Г. А. Звенигородский —

*Первоначальная версия этого материала была опубликована в ротапринтном сборнике ВЦ СО АН СССР (Новосибирск, 1986).

инженер одного из харьковских конструкторских бюро. Однако то, что в студенческие годы было дополнением к учебе, постепенно становится главным делом жизни. Работа в школе юных кибернетиков приобретает систематический характер, ее опыт освещается не только в новогодних корреспонденциях [13], но и в предметных статьях, формулирующих долгосрочные задачи компьютерного образования [10, 14].

В 1976 г. произошло предопределившее дальнейшую судьбу Г. А. Звенигородского его знакомство с коллегами из новосибирского Академгородка. Здесь к этому времени сложилась группа активистов, желавших исследовать перспективы применения ЭВМ в школе, открывшиеся в связи с появлением мини- и микро-ЭВМ и систем коллективного пользования. В нее вошли новосибирский преподаватель физики С. И. Литерат, барнаульский учитель Э. Ю. Форер, преподаватель Н. А. Садовская, инженер Г. А. Плотников и автор этих строк. «Мотором» в этой группе был Ю. А. Первин, незадолго до этого переехавший в Академгородок из Горького, где он проводил опыты по обучению младших школьников основам программирования на основе специально придуманного языка «Школьник» (Боброва Т. В., Бутенко Р. В., Первин Ю. А. Уроки программирования в пятом классе // Математика в школе. 1974. № 6. С. 57—58). Геннадий Анатольевич написал Ю. А. Первину, и тот пригласил его в Новосибирск на небольшой сбор учеников и преподавателей, открывший счет известным впоследствии всесоюзным школам юных программистов. Здесь Геннадий Анатольевич впервые познакомился и принял на вооружение новое словосочетание «школьная информатика», ставшее теперь уже привычным.

Планы работы новосибирской группы школьной информатики увлекли Звенигородского, и он, привезя снова группу харьковских школьников на вторую Всесоюзную школу программистов в июне 1977 г., через два месяца приехал в Академгородок снова, чтобы начать учебу в аспирантуре Вычислительного центра.

Деятельность Г. А. Звенигородского в Академгородке приобрела поистине титанический характер. Помимо индивидуальной научной работы он вел воскресную и вечернюю школы юных программистов, сотрудничал с общественной лабораторией проектирования прикладных программ, участвовал в руководстве и проведении заочной школы юных программистов, организовывал учебный процесс в ежегодных всесоюзных летних школах юных программистов, руководил проектными работами по договорам на разработку учебного программного обеспечения, был ответственным исполнителем по теме государственного плана и международного сотрудничества по линии СЭВ, вел занятия со школьниками на экспериментальных уроках по информатике в школе № 166 и в межшкольном учебно-производственном комбинате Советского района Новосибирска, проводил летнюю практику школьников в Вычислительном центре СО АН. Обеспечить дисциплину

и соблюдение техники безопасности оравой нескольких десятков подростков, наполнявших коридоры и терминальные комплексы института,— только учителям понятно, какая это была дополнительная нагрузка.

Вся эта разнохарактерная деятельность подкреплялась неустанными размышлениями, обстоятельной подготовкой к сдаче экзаменов кандидатского минимума, непрерывной литературной работой, бесконечными обсуждениями вопросов информатики с коллегами. Фигура Геннадия Анатольевича, неторопливо прогуливавшегося с очередным собеседником вдоль коридора отдела информатики, в течение восьми лет была наиболее характерной деталью любого рабочего дня у нас на этаже.

Как известно, основными результатами работы Г. А. Звенигородского как разработчика программного обеспечения стали языки Робик и Рапира, графический пакет «Шпага». К концу его учебы в аспирантуре эти системы были реализованы в макетном исполнении на БЭСМ-6 и достаточно широко апробированы во внеклассной работе. Этот материал был аккуратно описан и представлен вниманию нашего ученого совета в виде кандидатской диссертации «Математическое обеспечение школьного учебного процесса» [37]. К большому удивлению и естественному огорчению аспиранта и его руководителя, эта диссертация, выдвинутая на защиту в январе 1981 г., не получила поддержки совета. Многое слишком оказалось непривычно: и строгий зал академического института, заполненный непоседливыми школьниками, и «фантастические» истории о невиданных способностях подростков к продуктивной работе на ЭВМ, и опасная близость педагогики к профессиональной работе. Добавим, что существовали предрассудки на счет детского производительного труда, посожалеем, что диссертант забыл, что он не в классе, а перед строгими судьями, и тон его был отчасти менторским, обратный эффект дал неумеренный восторг руководителя — другими словами, это была не столь уж редкая судьба незаурядной работы, опережающей общественное сознание.

Эта история отняла немало душевных сил у Геннадия Анатольевича, но одновременно и закалила его, укрепив решимость двигаться дальше в избранном направлении. Кроме того, дело не только сталкивалось с препятствиями, но и располагало поддержкой немалого числа дальновидных людей. Начав работать с осени 1980 г. младшим научным сотрудником Вычислительного центра, Геннадий Анатольевич возглавил работы по созданию интегрированной системы программирования «Школьница» для персональной ЭВМ «Агат». Эта тема решением ГКНТ и Госплана СССР была включена в одиннадцатый пятилетний план важнейших работ, предвосхитив тем самым последовавшее развитие школьной информатики.

Все упомянутые выше формы работы сохранились на 80-е годы (за исключением заочной школы юных программистов, на повторение которой у «Кванта» не хватило пороху) и только получили больший размах. Летние школы юных программистов собирали теперь по 200 учащихся со всего Советского Союза с постоянным участием небольших «команд» из ГДР, Чехословакии, Болгарии, Венгрии. В университете сложилась группа студентов, прошедших выучку в районной школе юных программистов и поэтому с первого курса активно подключившихся к продуктивной научно-конструкторской работе. Они составили основу авторского коллектива системы «Школьница», опытная эксплуатация которой началась в 1984 г. [48]. Работа группы школьной информатики все в большей степени начала приобретать всесоюзный характер с установлением научных контактов с единомышленниками в Барнауле, Красноярске, Владивостоке, Свердловске, Тбилиси, Москве, Ленинграде, Таллине, Тарту, Вильнюсе, Кишиневе, Киеве и, конечно же, в родном Харькове [28].

Успешное завершение работ по системе «Школьница», ее широкое признание специалистами, документы по школьной реформе, общий ход времени вывели работу группы школьной инфор-

матики на уровень государственной деятельности. Сильно вырос и Г. А. Звенигородский: его богатый практический опыт был к этому времени подкреплен глубокими научными проработками программирования как вида деятельности, роли языков программирования и программных средств в учебном процессе. В июне 1984 г. он представляет к защите новую диссертацию «Система программирования, ориентированная на учебный процесс». Звенигородский приближается к ощущению весомости и признанию своей работы. По теме диссертации опубликовано 15 работ, его книга «Первые уроки программирования» [51] принята издательством «Наука», книга «Вычислительная техника и ее применение» — издательством «Просвещение», из разных мест приходят акты о внедрении системы «Школьница», получены первые, весьма положительные отзывы оппонентов, организация кабинетов вычислительной техники в соответствии с документами по школьной реформе включена в государственный план.

Трудно примириться с тем, что судьба распорядилась иначе. 8 октября 1984 г. на 33-м году жизни, подхватив банальную простуду, перешедшую в нераспознанное вовремя острое воспаление легких, Геннадий Анатольевич Звенигородский скончался за девять дней до назначенного срока защиты его диссертации.

Звенигородский хорошо понимал, что никакое, даже самое успешное педагогическое новшество не станет реальностью, если оно не войдет в сознание широкой массы учительства. В связи с этим большое место в его работе занимает пропагандистская и литературная деятельность. Его перу принадлежат корреспондентские повествования и отчеты о разных формах работы с учащимися: уже упоминавшиеся выше заметки харьковского периода, серия репортажей о III-й [21], IV-й [27] и VI-й [36] всесоюзных летних школах юных программистов, небольшая заметка во французском журнале [49].

Другой пласт работ Звенигородского составляют учебно-методические материалы, которые, по-моему мнению, весьма интересны для исследователя как образец конкретного педагогического материала, не обремененного излишними рассуждениями, но содержащего немало конкретных находок, из которых впоследствии вырастали более общие концепции и установки. Это описание Кибика [12], предвестника Робика, три части методического пособия по основным понятиям программирования [15], сложившимся еще в харьковский период, но навсегда вошедшим в методические установки Звенигородского; получивший широкую известность цикл публикаций по Рапире и «Шпаге» и основам программирования для заочной школы программирования на страницах «Кванта» [22—26, 31—35, 41, 42] и, наконец, относительно недавний методический материал, уже отражающий опыт преподавания в условиях компьютерного кабинета [44].

Из более наукоемких публикаций следует, конечно, в первую очередь назвать известную работу, вышедшую в 1979 г. в виде препринта ВЦ СО АН [17]. Эта работа, ставшая одновременно и манифестом, и программой работ сибирской группы школьной информатики. Не желая искусственно преувеличить роль этой публикации, хотел бы тем не менее сказать, что для нас является источником глубокого удовлетворения степень «угаданности» общих направлений, по которым сейчас развивается школьная информатика.

Важным достоинством работы Г. А. Звенигородского было стремление, несмотря на азарт и трудоемкость практической деятельности, осмыслить и выразить в общей форме хотя бы промежуточные итоги исследования. Начальный период работы 70-х гг. нашел свое отражение в сжатой статье в журнале «Математика в школе» [40]. Более обстоятельные статьи появились в двух международных изданиях: канадском журнале «Alberta Printout» [46] и трудах международной конференции в ГДР [50].

Важной догадкой Г. А. Звенигородского явилось подкрепленное опытом убеждение, что разработка программного обеспечения школьного учебного процесса по разным предметам должна

стать делом самих школьников, воплощением их опыта в изучении информатики и профессиональной ориентации. Ряд его публикаций раскрывает эти положения: первые варианты технических решений [16, 18], опыт, подкрепленный рядом экспериментальных работ [45], учебно-методологические вопросы профориентации и разработки учебных пакетов [54].

В наиболее полной форме итоги научной работы Г. А. Звенигородского по программному обеспечению школьного учебного процесса отразились в его диссертации 1984 г. [47] и в публикациях, составивших ее основу.

Г. А. Звенигородский рассмотрел программирование как деятельность и распространил известные две формы программирования («для себя» — интровертированная, «по заказу» — экстравертированная) на учебное программирование, выделив его четыре формы: учебно-интровертированную, учебно-экстравертированную, учебно-общекультурную и учебно-универсальную. На основе этой классификации сложился общий подход к системному анализу форм и методов применения вычислительной техники в учебном процессе [43].

Пользуясь этой методикой целевого системного анализа, Г. А. Звенигородский подверг подробному разбору наиболее известные языки, используемые в разных странах в учебном процессе [39]. Им было убедительно показано, что, невзирая на заметное разнообразие используемых языков, проблемы, здесь имеющиеся, далеки от своего решения. Тем самым разработка известных языков Рапира и Робик получила солидное научное обоснование.

Очень важной и еще не исчерпанной по своим ресурсам стороной работы Геннадия Анатольевича была отработка системно-программных технических решений по реализации учебного программного обеспечения с учетом особенностей архитектуры персональных ЭВМ, систем коллективного пользования и диалогового характера в компьютерном кабинете. Здесь и общесистемные вопросы [19, 30], и номенклатура компонент программного обеспечения [20], описания отдельных компонент [29, 38], наконец, первое сводное описание системы «Школьница» [48].

Отдельно следует отметить посмертные публикации Г. А. Звенигородского, излагающие, по существу, положения основных глав диссертации, подготовленных к печати его товарищами [52, 53].

Мы еще ничего не сказали о первой публикации Г. А. Звенигородского [1]. Это романтическое стихотворение, которое называется «Память», о летчике, поднявшемся во имя долга навстречу своей гибели в последний бой. Оно кончается необычной для подростка сильной и остро воспринимаемой сейчас рефлексией:

«... Это значит, что сильнее
смерти
Память.
И пока она живет,
Летчик будет где-то,
В чьем-то сердце
Продолжать свой
прерванный полет».

Память о Г. А. Звенигородском, одном из лучших представителей современной молодежи, будет жить не только в сердцах знавших его людей, но прежде всего в том деле, становлению которого он отдал без остатка все свои жизненные силы.

Список опубликованных трудов Г. А. Звенигородского

1967

1. Пам'ять // Ленінська зміна. 1967.15 мар.

1974

2. Построение оптимального типажа // Управление сложными технико-экономическими системами. Киев, 1974 (совместно с А. В. Дабагяном).
3. Знайди свій шлях // Ленінська зміна. 1974. 8 окт.
4. Фізика, націлена у завтра // Ленінська зміна. 1974. 22 окт.
5. Як зробити машину? // Ленінська зміна. 1974. 26 нояб.

1975

6. На межі фантастики // Ленінська зміна. 1975. 9 янв.

7. Олімпіада юних кібернетиків // Ленінська зміна. 1975. 29 мая.

8. Олімпіада кібернетиків фінішувала // Ленінська зміна. 1975. 26 июня.

9. Школа юних кібернетиків // Ленінська зміна. 1975. 6 сент.

1976

10. Программа первого курса школы юных кибернетиков на 1976/77 учебный год. Харьков, Б. И., 1976.

11. Моделирование макроструктуры сетей в системах обработки и отображения информации // Перспективные системы передачи, обработки и отображения информации. 1976 (совместно с В. В. Калугиным).

12. Кибик идет по Марсу // Пионер. 1976. № 10.

1977

13. «Кібік» крокуэ Марсом // Вечерній Харків. 1977. 1 янв.

14. Из опыта работы Харьковской школы юных кибернетиков // Углубленное изучение математики и ее приложений. М., 1977. С. 73.

1978

15. Основные понятия программирования: Методическое пособие, Новосибирск, 1978. Ч. 1—3.

16. Организация прикладных программных комплексов, ориентированных на обеспечение учебного процесса // Вычислительные системы, сети и центры коллективного пользования: Тр. Всесоюз. конф. Новосибирск, 1978. С. 111—113.

1979

17. Школьная информатика: концепции, состояние, перспективы. Новосибирск, 1979. С. 51 (препринт / АН СССР, сиб. отд., ВЦ; 152) (совместно с А. П. Ершовым, Ю. А. Первиным).

18. Структура математического обеспечения человеко-машинной обучающей системы в средней школе // Человеко-машинные обучающие системы: Тез. докл. I Всесоюзн. конф. М., 1979. С. 110—112.

19. Диалоговые компоненты системы математического обеспечения, ориентированной на школьный учебный процесс // Интерактивные системы: Тр. Советско-финского симпозиума. М., 1979. С. 410—421.

20. Компоненты системы математического обеспечения, ориентированной на школьный учебный процесс // Новые задачи информатики. Новосибирск, 1979. С. 121—131.

21. III летняя школа юных программистов // Программирование. 1979. № 3. С. 90—91.

22. Зачем надо учить программировать // Квант. 1979. № 9. С. 47—51 (совместно с А. П. Ершовым).

23. Законы программирования: Правила записи предписаний на языке Робик // Квант. 1979. № 9. С. 52—54.

24. Гибкие системы предписаний, синтаксические диаграммы и переменные поля // Квант. 1979. № 9. С. 55—57.

25. Работа с памятью. Имена и их значения // Квант. 1979. № 10. С. 54—57.

26. Арифметические предписания языка Робик. Условные и циклические предписания. Процедуры с параметрами // Квант. 1979. № 11. С. 44—47.

27. IV летняя школа юных программистов // Квант. 1979. № 12. С. 37—44 (совместно с Л. Н. Виленкиной).

28. Здесь учат машины // Комсомольское знамя. 1979. 12 дек.

1980

29. Машинная графика в математическом обеспечении учебного процесса // Прикладные методы информатики. Новосибирск, 1980. С. 121—133 (совместно с А. К. Салиховой, В. А. Цикозой).

30. Система математического обеспечения, ориентированная на школьный учебный процесс // Управляющие системы и машины. 1980. № 5. С. 76—82.

31. Основные операторы учебно-производственного языка Рапира // Квант. 1980. № 1. С. 52—55.

32. Описание и вызов процедур на Рапире // Квант. 1980. № 2. С. 53—55.

33. Функции, графики, локальные имена и тексты на Рапире // Квант. 1980. № 3. С. 48—51.

34. Множества и кортежи на Рапире // Квант. 1980. № 3. С. 51—56.

35. Обработка текстов на ЭВМ // Квант. 1980. № 11. С. 42—44.

36. V Всесоюзная летняя школа юных программистов // Квант. 1980. № 12. С. 40—51 (совместно с Ю. А. Первиным, А. Б. Сосинским).

1981

37. Математическое обеспечение школьного учебного процесса (средства формирования операционной обстановки): Автореф. дис. ... канд. физ.-мат. наук. Новосибирск, 1981. 23 с.

38. Язык начального обучения Робик в учебной системе программирования // Программное обеспечение задач информатики. Новосибирск, 1981. С. 72—85.

39. Некоторые особенности учебных языков и систем программирования // Экспериментальная информатики. Новосибирск, 1981. С. 81—91.

40. Работа со школьниками в области информатики: Опыт Сибирского отделения АН СССР // Математика в школе. 1981. № 1. С. 47—50 (совместно с А. П. Ершовым, С. И. Литератом, Ю. А. Первиным).

41. Работа с графами // Квант. 1981. № 3. С. 57—59.

42. Что такое мини-ЭВМ // Квант. 1981. № 6. С. 53—55 (совместно с Е. П. Кузнецовым).

1982

43. Некоторые вопросы методологии учебного программирования // II Чехословацко-советский семинар молодых ученых по математической информатике: Тез. докл. Братислава, 1982. С. 44—50.

1983

44. Методические рекомендации: Основы программирования. Новосибирск, 1983. Ч. 1,2 (совместно с Н. А. Юнерман).

45. Учебный пакет прикладных программ для школьных вычислительных кабинетов // Автоматизация производства пакетов прикладных программ: Тез. докл. II Всесоюз. конф. Таллин, 1983. С. 166—167.

46. Computers in School: Formulating a National Program // Abelrta Printout. 1983. V. 4. N 2. P. 34—37 (ut. auth. A. Ershov, U. Pervin, N. Yunerman).

1984

47. Система программирования, ориентированная на школьный учебный процесс: Автореф. дис. ... канд. физ.-мат. наук. Новосибирск, 1984. 14 с.

48. Программная система «Школьница» и ее реализация на персональных ЭВМ // Микропроцессорные средства и системы. 1984. № 1. С. 50—54 (совместно с Н. Г. Глаголевой, П. А. Земцовым, Е. В. Налимовым, В. А. Цикозой).

49. Informatique chez les enfants siberiens // Education 8 с informatique, 1984. N 19. P. 47—48. (It. auth. U. Pervin, N. Junerman).

50. ЭВМ в школе: опыт формирования национальной программы // INFO-84, 1984, 6—10 Februar, Drezden. Plenarvortrage, Teil 1, s. 53—63 (совместно с А. П. Ершовым, Ю. А. Первиным, Н. А. Юнерман).

1985

51. Первые уроки программирования. М., 1985. 208 с.— (Б-чка «Квант». Вып. 41).

52. Анализ форм программирования и способы их сочетания в программных системах, используемых в школьном учебном процессе // Автоматизированные рабочие места интеллектуальной деятельности: Сб. науч. трудов. Новосибирск, 1985. С. 4—24.

53. Реализационные особенности системы «Школьница» // Автоматизированные рабочие места интеллектуальной деятельности: Сб. науч. трудов. Новосибирск. С. 25—48 (совместно с Н. Г. Глаголевой, П. А. Земцовым, Е. В. Налимовым, В. А. Цикозой).

1986

54. Профориентационное обучение программированию в условиях школьного кабинета информатики (из опыта работы). Новосибирск, 1986. С. 21 (препринт / СО АН СССР, ВЦ; № 646) (совместно с Ю. А. Первиным).

Наталья Глаголева, студентка НГУ:

Геннадий Анатольевич — человек, который занимался своим делом. Знал то, что делал. Работал с полной отдачей. Горел. И — очень добрый. Такие люди всегда привлекают. Таких немного.

Нас многому научил. Гораздо больше, чем просто программированию. Это было первое место для меня, где увидела людей работающих, увлеченно и радостно работающих. Подарил нам чувство ответственности. В VI—VII классе чувствовать, что отвечаешь за все, что происходит на летней школе,— это серьезно. А доверить VI—VII классу организацию хотя бы части мероприятий ЛШ, решиться на такое — это не каждый может. Спасибо, что доверил.

Очень любила, как он читал стихи. Собственно, с тех пор и полюбила стихи.

При всех своих достоинствах — и мелких недостатках — предельно скромный человек.

Не могу о нем — в прошедшем времени.

Ева Елинер, студентка НЭТИ:

В нашей группе УПК ребята занимались разработкой обучающих программ чуть ли не по всем предметам курса, от физики до русского языка. И всем требовалась помощь: совет, подсказка. Звенигородский знал все. Переходя от машины к машине, за которыми мы работали, он с легкостью переходил от одной темы обсуждения к другой. Каждый предмет он знал, кажется, так, как будто бы по меньшей мере всю жизнь его преподавал. Он спорил о химии с лучшими химиками нашего класса и объяснял нам, девочкам, те разделы физики, которые мы еще не проходили, с терпеливостью учителя начальных классов.

Он любил свое дело, своих учеников, огорчался из-за наших неудач, радовался успехам, но больше любых результатов Геннадий Анатольевич радовался умным и интересным вопросам и всегда отвечал на них с воодушевлением. Он допускал, что можно чего-то не понимать, но как можно не хотеть понять?

Помню, Звенигородский часто нам говорил: «Думать надо! Думать!»

Юнерман Н. А., Новосибирск:

В нем было много таких черт, которые меня всегда поражали. Мог работать сутками, при этом очень много читал самой разной литературы. И, что удивительно, все помнил и всегда мог к слову рассказать соответствующий эпизод. Он очень многих приобщил к поэзии. С каждым (от ученика до академика) разговаривал на его уровне, уважал в собеседнике его знания и умения. И хотя сам мог очень многое, но уважал в человеке только ему присущие черты и его работу. Он научил меня программированию в самом широком смысле слова (по специальности я — учитель математики), он был моим учителем и коллегой, одновременно.

Неповторимо вел занятия с ребятами. Многие из них, работая сейчас со школьниками, пытаются использовать его приемы, но это не всегда возможно, они ловят себя на мысли, что так, как у него, у них не получается. Он и лекции по технике безопасности читал так, что никто сейчас не может читать их после него.

Не только великолепно вел занятия — умел и описать их захватывающе. Очень жаль, что успел написать так мало. Попытки дописать его книгу в его стиле нам не удались, поэтому добавлено только краткое описание Рапиры в виде приложения.

Не только учил ребят, он их воспитывал, ненавязчиво, без длинных нотаций, просто своим поведением. Был очень вежлив, его «здравствуйте», «извините», «пожалуйста» даже иногда раздражали, но были неотделимы от него. Бесконечно честен и принципиален. Очень многими своими хорошими манерами наши мальчишки обязаны ему.

Он умел радоваться успехам других.

Штернберг Л. Ф., г. Куйбышев:

Я работал с ним вместе в основном только на летних школах, поэтому наиболее яркие впечатления остались именно со школ.

Кроме программирования есть на школах еще что-то, что можно назвать духом школы. И этот дух школы во многом создан Геннадием Анатольевичем. Сколько он делал всего, мы поняли только на 10-й юбилейной школе, которую впервые проводили без него. Кроме учебной работы он проводил литературные вечера, где читал ребятам фантастику и где яблоку негде было упасть; участвовал в организации костров, экскурсий и прочего. С летних школ осталось огромное количество «программистских» песен, сочиненных для костров, и во многих из них с теплотой отзываются о Геннадии Анатольевиче.

Нас всегда восхищает, когда кто-то умеет делать то, чего не умеем делать мы сами. В особенности, когда это относится к сфере твоей профессиональной деятельности. Придумать тему, которую можно предложить для разработки на летней школе, сложно. Она должна быть достаточно простой, чтобы школьникам было под силу ее реализовать; в пределах школьного курса; с очевидной полезностью; чтобы на дисплее выглядело красиво и чтобы диалог был понятный и т. д. Я видел, как это делает Звенигородский: 5 минут беседы со школьником, которого он впервые видит,— выискивается тот предмет, который школьник лучше всего знает,— и тут же рождается тема в этой области, нужного уровня сложности.

Талант ученого и талант учителя — это разные и совершенно не зависящие друг от друга вещи. Ученый ищет, как сделать то, что пока никто не знает, как сделать; учитель ищет, как донести до новичка то, что известно всем,— и это не проще, чем найти новое. Звенигородский сочетал и то и другое.

Титов О. Ф., Москва:

Гену Звенигородского я знал и не знал. Когда встречались — тут же возникала необходимость обсудить какие-нибудь срочные и архисрочные дела, у нас было мало времени познакомиться с внутренним миром друг друга.

Думаю, что был он прежде всего педагогом. Человеком сложным, но цельным. Работал плотно, без зазоров — не все энтузиасты смогли с ним работать. Для меня решающими были его отношение к детям и полная самоотдача.

Сам был тружеником и умел завести других. Его умение помогать детям ставить себе задачи восхищало многих. На первый взгляд банальные, либо архисложные, либо вообще далекие от программирования — они будили в детях вулканы фантазии, парадоксальных подходов, интересных решений. По-моему, на летней школе он тратил больше сил на то, чтобы отвлечь детей от компьютера (а ведь они не в игры играли!). Именно на его летних школах были театрализованные представления, факультатив танцев, литературные вечера, интересные спортивные матчи, стенные газеты и свой необъятный песенный фольклор.

Считаю, что программирование было для него прежде всего средством воспитания. Возникающее часто еще в школе отношение к любой работе, как к насилию над собой, разрушает в нас человека. Подход Звенигородского заключался в том, чтобы привлекать детей как можно раньше к участию в настоящем деле, дать им почувствовать вкус к самовыражению в коллективной работе с общественно полезным результатом. Программирование, по-моему, просто более всего подходило ему как достаточно редкая возможность разрабатывать законченный продукт (от идеи до полной реализации) в рамках небольшого коллектива, последовательно решая творческие задачи (от простых до увлекательнейших головоломок).

В итоге им (с учениками) создан не только известный учебно-производственный язык Рапира, результат выдающийся, но и уникальная учебно-производственная среда со своими традициями, коллективом.

Думаю, он занимал достойное место среди наших педагогов-новаторов.

М. ГОТЛИБ

Структура АОС

Мотивация — важнейшее условие эффективности обучения. Повышает ли ее применение вычислительной техники? Несомненно. Однако графические, цветовые и динамические возможности ПЭВМ, наглядность изображения материала могут влиять лишь на внешнюю мотивацию. При всем своем положительном значении она остается формальной, оторванной от предметного содержания. Формирование же внутренней мотивации в процессе обучения и, следовательно, подлинного интереса к предмету открывает для развития мышления огромные возможности. С этих позиций и рассмотрим вопросы создания автоматизированной обучающей системы (АОС).

Компьютер как средство обучения можно использовать для:

демонстрации законов и явлений природы;

моделирования процессов;

тренировки с целью выработки умений и навыков;

работы с прикладными программами, обрабатывающими текстовые и арифметические выражения, информационные массивы, результаты лабораторных и практических работ и т. д.;

контроля знаний, умений, навыков;

обработки результатов обучения с целью определения психологических особенностей и уровня подготовленности обучаемого;

выработки стратегии оптимального обучения индивида;

собственно обучения.

Первых два направления хорошо известны; их реализация основывается преимущественно на графических возможностях ПЭВМ; создание программного обеспечения серьезных трудностей не представляет. Работа в режиме демонстрации и моделирования характеризуется низкой степенью индивидуализации, которая сводится прежде всего к управлению самим обучаемым демонстрируемыми процессами и явлениями — изменению их параметров, установлению временных и пространственных режимов, степени детализации и т. д.

Использование ПЭВМ в качестве тренажера предполагает более сильную обратную связь. Программа обучения составляется в зависимости от достигнутых учащимся успехов. Как только они начинают соответствовать нормативным показателям, система переключается на отработку следующего элемента. Тренажер используется, как правило, для формирования и развития начальных навыков. Эта фаза обучения наиболее утомительна и трудоемка. Рационально организовав ее в виде игры, мы тем самым устанавливаем определенную психологическую связь между игровой ситуацией и познавательной деятельностью, что на первом этапе повышает лишь внешнюю мотивацию. Но, как показывают педагогические наблюдения, постепенно осуществляется перенос мотивации из внешней формы во внутреннюю. Мотивированным становится сам процесс познания. Эта тенденция подкрепляется и осознанием учащимся того, что самое трудное позади. Навыки накоплены, появляется потребность в их реализации. Обучение становится по-настоящему интересным. Формальная игра превращается во внутрипредметную, уже не требуется внешний «допинг».

На практике, конечно, не все так гладко. Но в тех случаях, когда разумно определен набор навыков, правильно установлены нормативы их усвоения, органически сочетается с ними игровая ситуация, а по мере приобретения навыки ориентируются на решение интересных и даже творческих задач, реальная ситуация удивительно напоминает идеальную модель.

Обратимся теперь к элементам предметной области, без которых немыслимо дальнейшее исследование возможностей автоматизированного обучения.

Опорные элементы

Центральное место в любой обучающей системе занимают опорные элементы знаний (ОЭЗ). Если они не разработаны методически подробнейшим образом, достичь высокой эффективности обучения невозможно. ОЭЗ образуют базис предмета и находят разнообразное и действенное применение на практике, если удовлетворяют некоторым принципам.

Одни и те же элементы лежат в основе изучения разных тем, а иногда и разделов предмета, поэтому каждый из них должен быть сформулирован так, чтобы с равным успехом использоваться в любом конкретном случае,— принцип *универсальности*. Любое предметное понятие должно опираться на некоторый набор ОЭЗ, полностью подготавливающих его восприятие,— принцип *полноты*. При этом каждый элемент в отдельности, хоть и связан с другими, не сводится к их простой комбинации — принцип *элементарности*. По понятным соображениям, ОЭЗ должны формулироваться предельно четко и лаконично — принцип *компактности*.

Формы преподнесения ОЭЗ различны: их можно, например, формировать в процессе демонстрации, при выполнении обучающих заданий, допустимо и пассивное ознакомление с последующей установкой на применение.

ОЭЗ несут большую функциональную нагрузку в АОС: они являются предметом самостоятельного изучения, методической основой при изучении комплексных понятий, элементами диагностики для определения психологических особенностей индивида, параметрами оптимизации процесса обучения и т. д.

В качестве примера рассмотрим использование ОЭЗ для решения кинематических задач по курсу физики средней школы.

Рассмотрим задачу: автомобиль и велосипедист, расстояние между которыми 250 м, движутся навстречу друг другу со скоростями 15 и 5 м/с; считая, что автомобиль в начальный момент времени проходит начало координат в направлении оси, найти точку, которую он проедет на 5 с раньше велосипедиста. Наша цель: не решать, а всего лишь записать условие задачи в виде формализованных соотношений, где и будет заключено решение. Вот как это можно осуществить.

Опорными для этой задачи являются следующие элементы.

1. Уравнение прямолинейного равномерного движения $x=x_0+v_xt$.

2. Если вектор направлен вдоль координатной оси, то его проекция равна модулю, $v_x=v$.

3. Если вектор направлен противоположно координатной оси, то проекция противоположна модулю, $v=-v$.

4. Условием нахождения тел в одной точке является равенство их координат ($x_1=x_2$).

Учащийся знает, что в уравнении движения (и вообще любого процесса) заключена вся информация о нем. Следовательно, прежде всего необходимо записать два уравнения движения. С использованием элементов 1 и 2 записывается уравнение движения автомобиля для момента его нахождения в искомой точке. Велосипедист же пройдет указанную точку спустя $\tau=5$, поэтому уравнение его движения с учетом элементов 1 и 3 записывают для момента $(t+\tau)$. Осталось отметить, используя элемент 4, что места нахождения обоих тел совпадают:

$$\begin{cases} x_1=v_1t \\ x_2=x_0-v_2t \\ x_1=x_2 \end{cases}$$

На этом физическая часть завершается. Решить систему уравнений и определить искомую координату помогут опорные элементы алгебры, но это уже другая тема.

С помощью приведенных элементов решаются всего несколько задач, зато, введя в общей сложности 12 элементов в кинематике и используя еще 5 из других разделов, можно решить большинство кинематических задач (во всяком случае, в пределах школьного курса физики).

Описываемый метод характеризуется высокой степенью алгоритмизации и формализации, следовательно, перед нами не только метод обучения чело-

века, но и одно из возможных направлений в обучении ЭВМ.

На вопрос: «Можно и нужно ли одним и тем же методом обучать и человека, и машину?» — ответим: обучать всегда нужно эффективными способами, и коль скоро мы доверяем машине обучение человека, то должны четко представлять себе и пытаться использовать все возможные плоскости соприкосновения, взаимопонимания человека и ЭВМ. Другой вопрос: «Не оказывает ли увлечение алгоритмизацией в обучении негативного влияния на развитие творческих способностей?» Нет. Предметные алгоритмы — мощное технологическое средство обучения. Вооружив ими учащихся, исключив слепые пробы и неизбежные ошибки, мы значительно повысим продуктивность обучения на его начальном этапе приобретения навыков и умений и, следовательно, создадим надежную базу для решения по-настоящему творческих задач в дальнейшем.

Обучающие задания

Каждый учитель-предметник в процессе усвоения материала помогает учащемуся, задавая наводящий вопрос, указывая на противоречия, к которым приводят допущенные ошибки, напоминая общее положение, которое необходимо применить к данной конкретной ситуации,— фактически использует обучающие задания. Иными словами, любую форму общения с обучаемым, требующую от того мыслительной активности и динамически управляющую актом усвоения им данного понятия, можно квалифицировать как обучающее задание. Отсюда следует, что одно и то же задание, выполняемое различными учащимися, даже если они находятся в тождественной обучающей среде, в одном случае может являться обучающим, а в другом — нет, в зависимости от того, повлияло ли оно на процесс формирования понятия.

Опишем кратко основные типы алгоритмов обучающих заданий. Они определяются порядком расположения обучающих элементов.

Последовательно-подготовительный. Начальный элемент задания относительно прост, он подготавливает выполнение второго, несколько более сложного, а тот, в свою очередь, третьего и т. д. Заключительные элементы имеют достаточно высокий уровень сложности (рис. 1).

Параллельно-подготовительный. Начальные элементы заданий независимо один от другого подготавливают выполнение следующего за ними комплексного элемента высокого уровня (рис. 2).

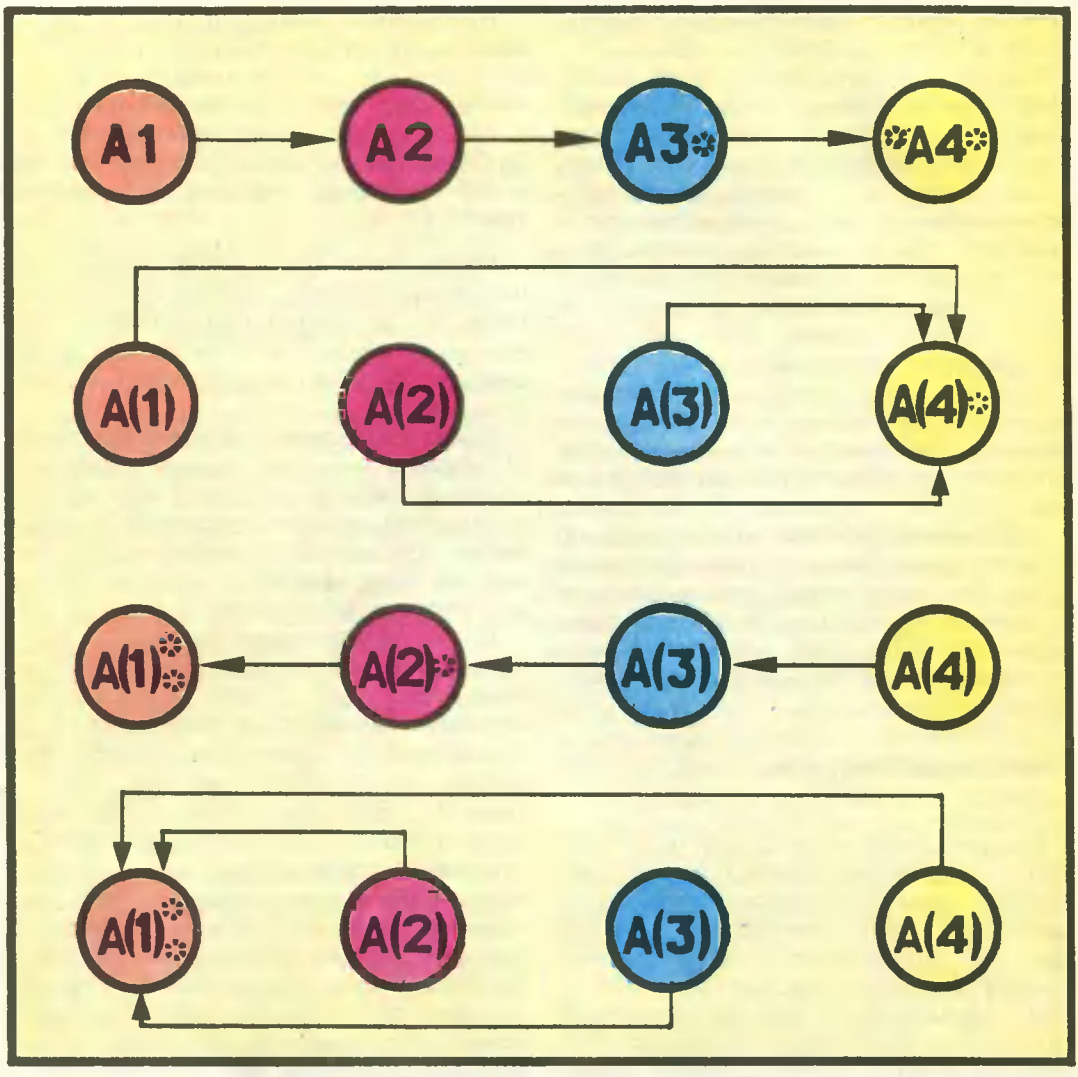
Последовательно - корректирующий. Начальные элементы задания имеют высокий уровень сложности, а каждый последующий элемент корректирует выполнение предыдущего, указывая, например, на противоречия, к которым приводят неправильные ответы (рис. 3).

Параллельно-корректирующий. Обучаемому предлагается комплексный элемент высокого уровня, последующие элементы играют роль наводящих (или подсказывающих), причем с разных позиций, независимо один от другого (рис. 4). Наблюдения за выполнением заданий такого типа приводят к выводу, что большинство учащихся после завершения работы вновь возвращаются к начальному элементу и исправляют ответ в случае, если первоначально он был неверным. Таким образом, задание трансформируется самим обучаемым в параллельно-подготовительное, однако с той особенностью, что выполняется оно с предварительной установкой.

В **алгоритме переноса** приводятся два массива элементов — A(M) и B(M). Ими могут быть понятия, отношения, действия, характеристики и т. д. Требуется установить логическое соответствие между ними (рис. 5).

В **аналитическом алгоритме** предлагаются элементы A(M). Необходимо установить принадлежность каждого из них к одному из классов B(K) (рис. 6).

Напротив, **в синтезирующем алгоритме** элементы массива A(M) уже разбиты на подгруппы A(M1), A(M2) и т. д. Задача обучаемого — установить критерий, по которому осуществлялась классификация (рис. 7).

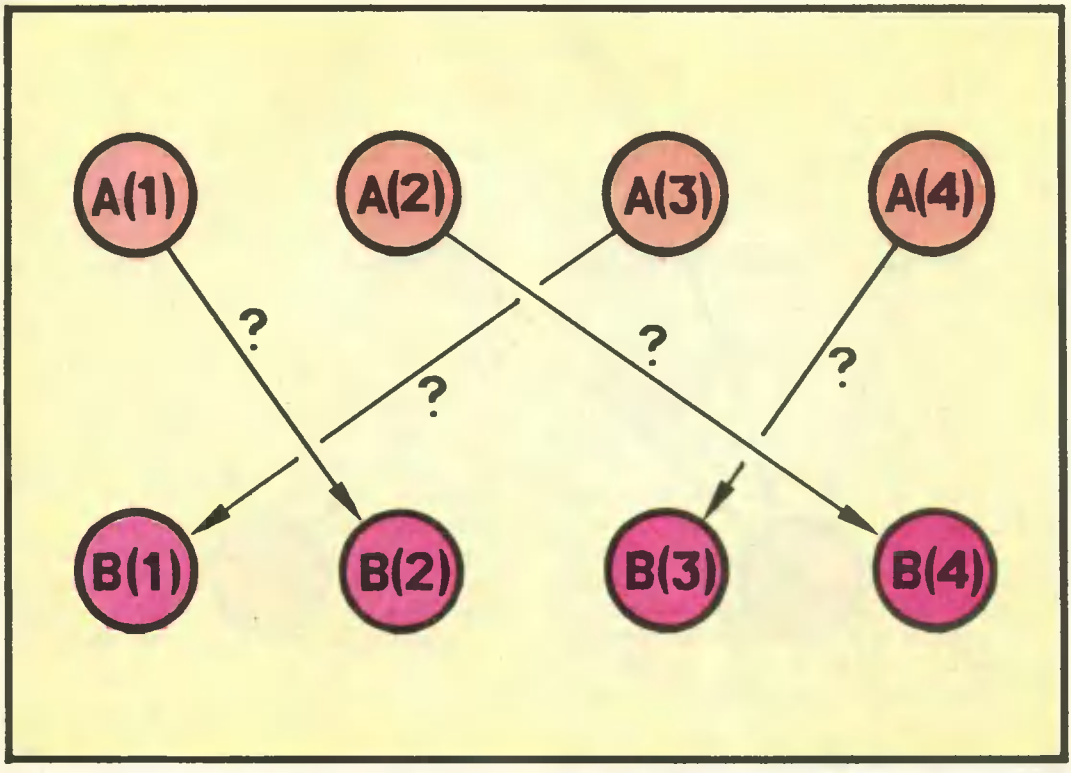


Алгоритм упорядочения элементов по некоторому указанному критерию требует для своего выполнения комплексной умственной деятельности.

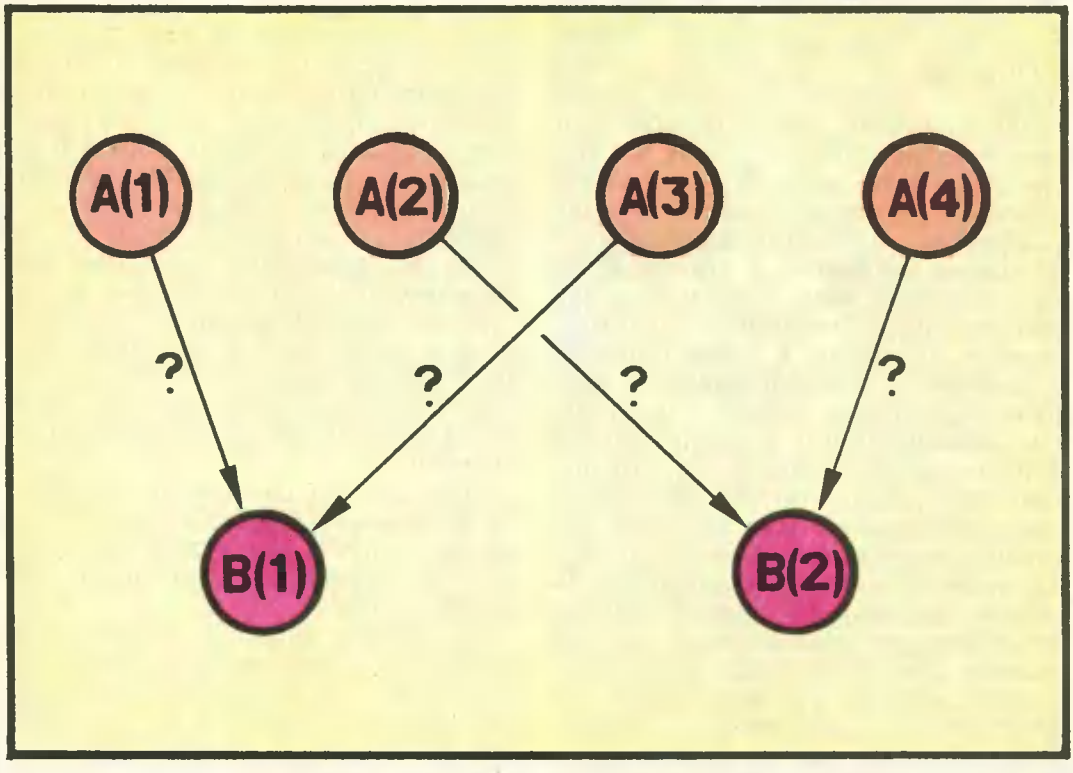
Алгоритм задания не следует путать с его формой, являющейся средством выражения этого алгоритма, оказывающей, кроме того, определенное психологическое воздействие на обучаемого, разнообразя его деятельность и тем самым повышая мотивацию обучения. Перечислим только основные выделенные нами формы: вставить пропущенные в тексте фрагменты; дать аналитическую (формализованную) запись данного текстового высказывания; прокомментировать аналитическую запись; завершить начатый фрагмент; расположить перепутанные фрагменты в правильной последовательности; провести аналогию; составить текст; установить истинность высказываний; ответить на вопросы; отредактировать текст; отработать (запомнить, а затем повторить; классифицировать) набор понятий в условиях органиченного времени и т. д.

Адаптивные функции АОС

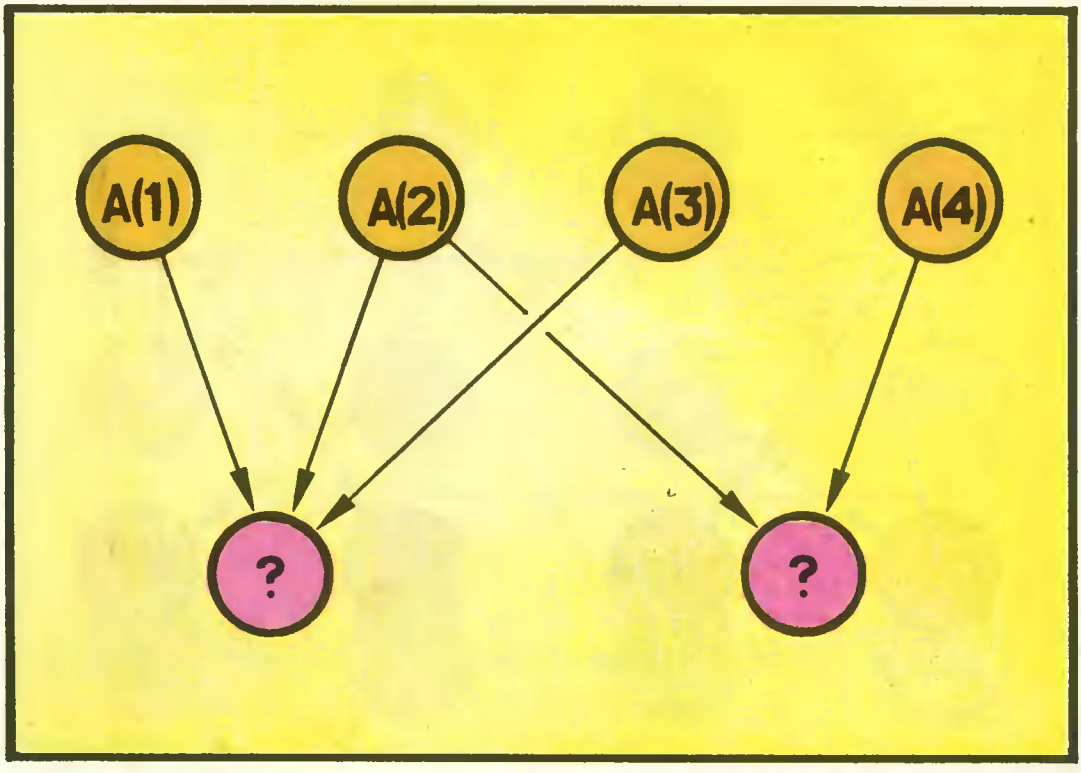
Важной особенностью АОС являются ее адаптивные свойства, возможность приспосабливаться к индивиду в зависимости от его психологических особенностей и достигнутых им успехов.



5



6



Рассмотрим эти свойства на примере вышеуказанных обучающих заданий. Можно выделить три основных направления адаптации: трансформация алгоритмов, нивелирование уровня сложности элементов заданий, установление объема и последовательности заданий.

Задания могут трансформироваться из одного алгоритма в другой. Изменяя, например, последовательность элементов, можно превращать подготовительные алгоритмы в корректирующие и наоборот, а изменив установку задания, осуществлять взаимные превращения сравнительного и избирательного алгоритмов. Для чего нужны эти превращения? Эксперименты, проводимые нами, показали: у разных учащихся наилучшие показатели в обучении достигаются при разных типах алгоритмов. Замечено, например, что имеющие хорошие аналитические показатели и высокий уровень подготовленности затрачивают меньше времени на выполнение корректирующих алгоритмов, показывают более глубокий уровень усвоения, в то время как группа с низкими аналитическими показателями лучших результатов добивается при работе с подготовительными алгоритмами. Это естественно. Как известно, имеются существенные вариации в ходе усвоения понятий, зависящие и от особенностей предметного характера, и от индивидуальных особенностей обучаемого. Часто это переход от конкретного к абстрактному, но возможен и другой вариант — от бессодержательно-абстрактного к конкретному и через него к подлинно абстрактному. Естественно, что «в резонансе» с каждым из этих вариантов находится свой тип или сочетание типов алгоритмов.

Для каждого обучаемого существует та единственная последовательность усвоения понятий и операций над ними, которая приносит максимальный обучающий эффект.

Не требует никакого доказательства положение о том, что сложность задания должна соответствовать уровню подготовленности учащегося. Вопрос в том, что понимать под уровнем развития и как этот уровень определить? Очевид-

но, решая задачи обучения, мы должны ориентироваться на зону ближайшего развития, т. е. опираться не на завершенный процесс развития, а на тот, который находится в становлении.

Можно снабдить АОС механизмами для трансформации типов алгоритмов, изменения последовательности заданий или выбора обучающих элементов, сложность которых соответствует определенному уровню, при том, однако, обязательном условии, что обучающая система располагает критериями всех этих адаптаций.

Таким образом, перед АОС встают еще две фундаментальные задачи: во-первых, осуществление психопредметной диагностики, включающей в себя определение психологических особенностей, уровня подготовленности индивида, специфику усвоения им предметного материала; во-вторых, оптимизация процесса обучения на основе этих данных.

Психопедагогическая диагностика

Психология всегда применяется в педагогической практике — в одних случаях на научной основе, в других — на интуитивном уровне, но, как правило, лишь в границах качественного подхода. Нужно ли в процессе обучения применять количественный анализ? Думается, да. Однако громоздкость и сложность делают невозможным даже эпизодическое его использование в современных условиях. Поэтому преподавателю не остается ничего другого, как оценивать результаты обучения поверхностными критериями. С появлением же ПЭВМ у педагога появляется возможность обучать, гораздо более точно оценивая динамику развития.

Какие индивидуальные показатели должны быть положены в основу такого подхода? Очевидно, не общепсихологические (интеллектуальные профили): они малоизменчивы и не могут описывать динамику усвоения предмета, которая более всего интересует педагога; кроме того, обучение конкретному предмету опирается не на фундаментальные интеллектуальные (например, обобщение), а на их приложение к узкой предметной области (например, обобщение механических понятий). Нужно ли доказывать, что это не одно и то же? Поэтому, на наш взгляд, искомыми характеристиками должны стать *психопредметные* профили, т. е. степень выраженности у индивида некоторых психологических особенностей, реализуемых при обучении данному предмету.

Вариант использования трех профилей — математического, естественного и гуманитарного — не упростит, а, скорее, усложнит решение задачи, возникнут проблемы координации и сочетания методик, создания межпредметных банков и т. д. Профилей должно быть столько же, сколько дисциплин изучает учащийся с помощью ПЭВМ. Перечислим основные психологические способности: полнота и устойчивость восприятия включают в себя сосредоточенное внимание, целенаправленность, волевые и другие черты характера; уровень развития процессов памяти — умение удерживать в сознании и использовать в своей деятельности общие схемы, правила, закономерности, методы, отношения и пр.; аналитические способности — умение представить комплексное понятие в виде составляющих его простых элементов; сравнение — умение сопоставить элементы, найти различие между ними и, как результат, выделить основное, отличить от второстепенного; синтезирующие способности — создание единого комплексного понятия из набора простых; трансформация понятия (гибкость, лабильность) — умение изменять второстепенные качества объектов, сохраняя их сущность (ядро) при использовании в новых условиях; аналогия — перенос логических схем и структур из одного пространства в другое; ассоциативность мышления — количество, широта и разнообразие связей, активизируемых в сознании при возникновении предметного раздражителя.

Выполнение большинства обучающих заданий основано на сочетании различных способностей, однако можно выделить группы заданий, которые преимущественно связаны с одной конкретной способностью. Такие задания, не теряя своих обучающих функций, могут одно-

временно приобретать и тестирующие. По результатам их выполнения можно судить о степени развитости тех или иных способностей.

Приведем возможный вариант использования психопредметного профиля в АОС: определение индивидуальных особенностей в процессе выполнения обучающих (они же тестовые) заданий; оптимизация процесса обучения с учетом выявленных характеристик; непрерывная регистрация результатов обучения и соответствующая коррекция психопредметного профиля и т. д.

Не будем сейчас останавливаться на вопросах, связанных с методами определения психологических особенностей, доказательством их надежности и валидности. Психология располагает почти столетним опытом разработок в этом направлении. В частности, факторный и таксономический анализ с некоторыми изменениями могут быть перенесены на почву предметной психологии с целью последующего использования в АОС. Например, обработка с помощью этих методов проводимых нами экспериментов позволила сделать следующие выводы: наблюдается корреляция между типами обучающих алгоритмов и изменением психопредметного профиля. Иными словами, подбирая соответствующим образом типы обучающих алгоритмов, можно изменять профиль обучаемого.

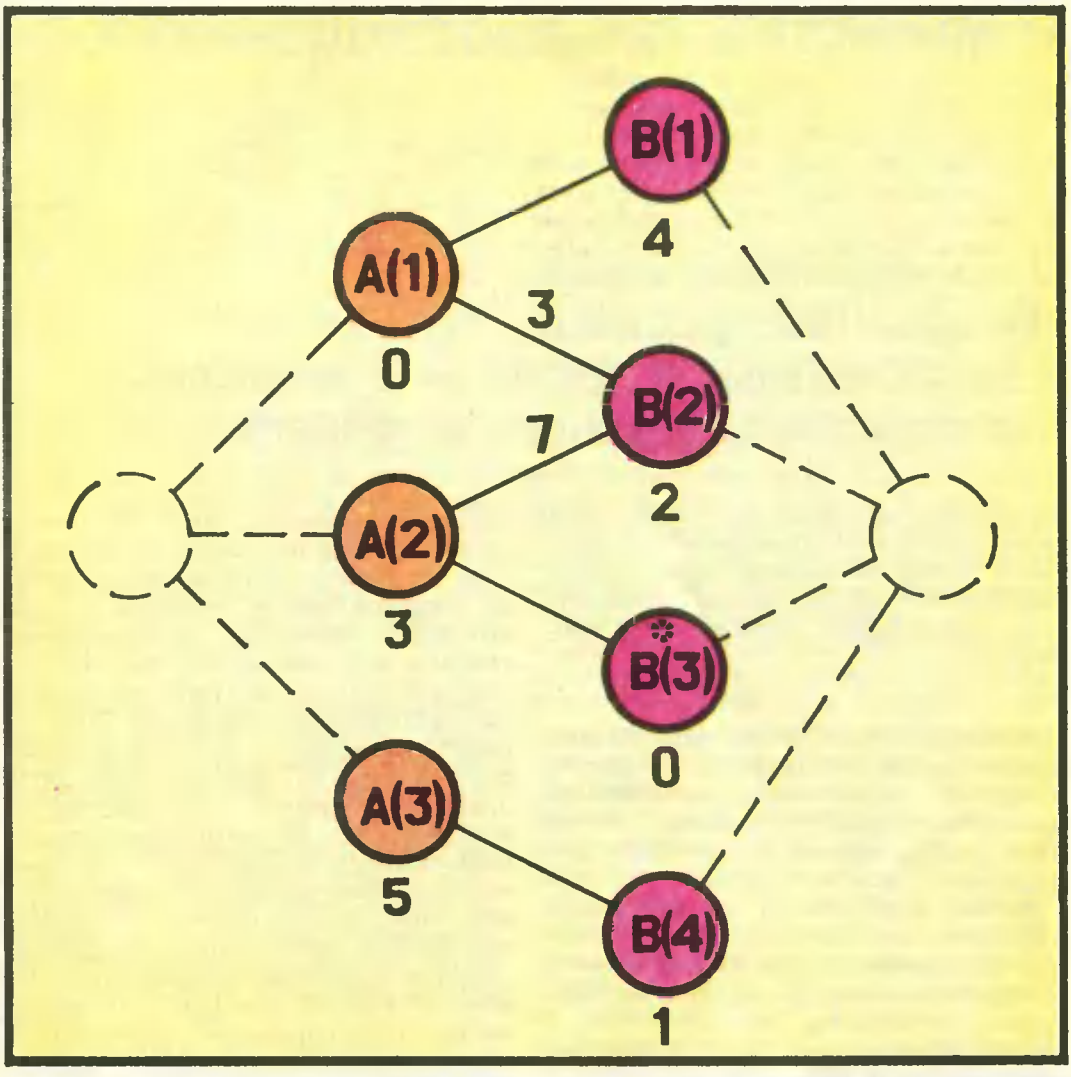
Выработка стратегии оптимального обучения

Так мы подошли к центральной проблеме АОС, усложненной еще и тем, что решать ее приходится на стыке таких областей, как предметная методика, психопедагогика, кибернетические методы обработки данных.

Оптимизируя процесс обучения, мы должны учитывать следующие параметры: психопредметный профиль; приоритет опорных элементов знаний, зависящий от конкретно-методической конструкции изучаемой темы и уровня подготовленности индивида; связь обучающего задания с опорными элементами знаний, которые отрабатываются в процессе его выполнения; приоритет алгоритмов обучающих заданий, зависящий от психопредметного профиля учащегося.

Один из возможных методов решения такой многопараметрической задачи — использование хорошо известных в кибернетике моделей оптимизации потока в сетях. Рассмотрим, например, задачу: зная психопредметный профиль обучаемого (и, следовательно, приоритет обучающих заданий), приоритет опорных элементов знаний и их связь с заданиями, оптимизировать процесс обучения.

Изобразим это графически (рис. 8). Пусть A(M) — массив опорных элементов знаний (M=1, 2, 3), а B(K) — массив обучающих заданий (K=1, 2, 3, 4). Под каждой вершиной A(M) и B(K) приводится значение ее приоритета. Связь задания с опорными элементами изображена с помощью ребер соответствующей длины. Задание B(2), например, отрабатывает опорные элементы A (1) и A(2) в эффективном отношении 3 к 7. Эти значения присвоены длинам ребер A(1)—B(2) и A(2)—B(2). Граф имеет инвариантный вид для всех обучаемых, а вот значения вершин индивидуализированы. Выбор программы обучения теперь можно свести к определению таких путей в графе, которые оптимизируют (минимизируют или максимизируют) некоторый показатель. Для учащегося с высоким уровнем подготовленности и хорошим психопредметным профилем следует, очевидно, максимизировать число заданий высокого уровня (в том числе и творческих). Для учащегося с низким уровнем подготовленности необходимо максимизировать поток обучения через вершины, соответствующие опорным элементам, в которых он проявил свою неосведомленность. Такие вершины в его графе будут иметь высокий приоритет (например, A(3)). Напротив, приоритет вершины A(1) равен 0, т. е. нормативный уровень владения соответствующим опорным элементом находится в зоне ближайшего развития индивида, и поэтому поток через эту вершину будет отсутствовать — задание B(1) игнорируется. Это, однако, не означает, что указанный элемент более не отрабатывается, ведь часть потока пройдет через ребро A(2)—B(2), а задание B(2)



частично отрабатывает и элемент А(1). Аналогично обнуляются вершины (В (3), например), соответствующие заданиям, сложность которых превышает уровень подготовленности обучаемого. Просчитывается и множество иных ситуаций.

В процессе работы повышается уровень подготовленности учащегося. Он переходит от одного критерия оптимизации к другому, выполняя каждый раз те задания, которые ему по силам и в то же время требуют мыслительной активности, а потому интересны. Все это готовит его к творческому решению сложных задач, которые поставит перед ним жизнь. Это, на наш взгляд, и есть социальный заказ общества в области компьютеризации.

Литература

Анастази А. Психологическое тестирование. М.: Педагогика, 1982.

Готлиб М. А. Обучающие и контролирующие функции программированных заданий по физике // Преподавание физики и математики. 1986. № 3.

Коршунов Ю. М. Математические основы кибернетики. М.: Энергия, 1980.

Крутецкий В. А. Психология математических способностей школьников. М.: Просвещение, 1968.

Стоунс Э. Психопедагогика. М.: Педагогика, 1984.

В Министерстве просвещения СССР

О преподавании курса «Основы информатики и вычислительной техники» с примерным планированием учебного материала

Это методическое письмо, составленное НИИ СиМО АПН СССР и Управлением информатики и электронно-вычислительной техники Минпроса СССР, рекомендовано к руководству и использованию в работе в 1987/88 учебном году.

Программа курса «Основы информатики и вычислительной техники» в зависимости от возможности организации практической работы школьников на ЭВМ предусматривает два варианта преподавания курса в X классе:

полный курс (68 ч) — для школ, располагающих вычислительной техникой или имеющих возможность организовать систематические занятия на ЭВМ других организаций;

краткий курс (34 ч) — для школ, не имеющих такой возможности.

Теоретическая часть курса X класса — единая для обоих вариантов; различие только в объеме и содержании практической части.

В школах, имеющих ЭВМ, дополнительные 34 учебных часа должны быть использованы для решения на ЭВМ различных задач, отработки навыков применения компьютера и его программного обеспечения.

В X классе практическую работу школьников на ЭВМ можно организовать в двух формах. Первая заключается в том, что компьютер с соответствующим учебным программным обеспечением используется при изучении информатики как для освоения теоретического материала, так и для практических занятий на каждом уроке. Ввиду того что учебная вычислительная техника сосредоточена в основном в межшкольных кабинетах вычислительной техники, а в школах ее пока недостаточно, широко используется другая форма организации работы на ЭВМ, а именно практикум в кабинете вычислительной техники, который проводится отдельно от теоретических занятий по информатике. Преимущественное использование этой формы ведения практических занятий обусловлено также тем, что для постоянного использования ЭВМ на каждом уроке необходимо развитое программное обеспечение. Поэтому в 1987/88 учебном году можно рекомендовать проведение практических занятий в основном в форме практикумов.

Программа курса допускает использование на практических занятиях любого имеющегося в распоряжении языка программирования. Учитывая наличие в действующем учебном пособии соответствующих разделов, предпочтение следует отдавать работе на Рапире или Бейсике.

Можно рекомендовать следующее примерное планирование практических занятий* в объеме 34 ч в кабинете вычислительной техники.

---

* Примерное планирование теоретической части курса приводится в методическом письме «О преподавании курса ОИВТ в текущем учебном году».

1. Начальные навыки работы с ЭВМ (2 ч)

Правила включения, загрузки и выключения ЭВМ, назначение ее основных устройств и демонстрация их работы. Инструкция по технике безопасности. Клавиатура ЭВМ. Самостоятельная работа с программой демонстрационного типа.

2. Исполнение алгоритмов на ЭВМ (8 ч)

Построение алгоритмов и их исполнение с использованием «Системы программирования на основе алгоритмического языка курса ОИВТ» (Е-практикум).

3. Знакомство с имеющейся системой программирования (Рапира, Бейсик и др.) (8 ч)

Средства создания, редактирования, запуска и отладки программы. Непосредственный и программный режимы. Задания на вычисления в обоих режимах (2 ч).

Особенности имеющейся реализации выбранного для работы языка программирования. Демонстрация работы его основных команд. Разбор и демонстрация работы готовых программ, написанных на данном языке программирования (2 ч).

Средства работы с файлами (1 ч).

Работа по реализации простейших алгоритмов (2 ч).

Самостоятельная работа № 1 (1 ч).

4. Прикладное программное обеспечение (12 ч)

Работа с простым редактором текстов (2 ч).

Работа с графическим редактором (2 ч).

Работа с учебной базой данных (2 ч).

Работа с учебными пакетами прикладных программ (3 ч).

Самостоятельная работа № 2 (1 ч).

Экскурсия на вычислительный центр для ознакомления с имеющимся там программным обеспечением (2 ч).

5. Решение учебных задач с использованием ЭВМ (4 ч)

Первые два занятия носят подготовительный характер. Учитель объясняет назначение и демонстрирует работу основных устройств ЭВМ, проводит инструктаж по технике безопасности. Далее следует самостоятельная работа школьников с простейшим программным обеспечением для получения начальных навыков работы с компьютером. В качестве программного обеспечения этих занятий можно порекомендовать программу, демонстрирующую те или иные возможности ЭВМ, клавиатурный тренажер. Компьютерные игры следует использовать осторожно, только для овладения навыками работы на клавиатуре. Работе с текстом и графикой будут посвящены отдельные занятия, однако элементарная работа этого плана возможна и на первых занятиях.

Цель, на достижение которой направлена серия из занятий третьего раздела, будет достигнута, если учащиеся научатся самостоятельно писать на выбранном языке программирования, отлаживать, пропускать на ЭВМ и записывать на внешние запоминающие устройства несложные программы (20—30 команд), использующие все основные конструкции языка. Если преподавание ведется на Рапире или Бейсике, то занятия опираются на соответствующие разделы теоретической части курса и служат их развитием. Сами алгоритмы, подлежащие программированию, можно брать из учебника готовыми или же предоставлять их составление учащимся. При разборе готовых программ следует отметить правила оформления программ, возможные ошибки, выделить основные языковые конструкции. Средства работы с файлами, которые упоминаются здесь, — это средства создания, записи, загрузки, уничтожения и переименования. Предусмотренная в этом разделе самостоятельная работа направлена на проверку полученных навыков программирования и должна содержать задание по реализации одного несложного алгоритма. При ее выполнении вполне допустимо использование справочных материалов, о чем надо предупредить учащихся заранее.

Содержание работы, предусмотренной в четвертом разделе практикума, в максимальной степени зависит от имеющегося программного обеспечения, и в случае, когда его не хватает, данный раздел сокращается, а освободившиеся часы распределяются между третьим и пятым разделами. Учебная задача, стоящая перед данным разделом, имеет два аспекта: во-первых, учащиеся должны освоить навыки работы с указанными программами, которые, как правило, являются не специфическими, а едиными для всего класса таких программ, а во-вторых, получить представление о тех задачах, для решения которых предназначены эти программные средства, и способах их решения. Самостоятельная работа, предусмотренная в этом разделе, представляет собой небольшую учебную задачу, предназначенную для решения с помощью одного из перечисленных программных средств.

Последний раздел позволяет организовать работу по решению содержательно поставленной задачи (из курсов математики, физики и др.) с использованием ЭВМ. Следует отметить, что имеется положительный опыт коллективной формы такой работы. Класс делится на группы, каждая из которых получает свою задачу. В процессе ее решения внутри группы происходит некоторое разделение труда, характерное для практических работ с использованием ЭВМ.

При возможности организовать практические занятия в IX классе рекомендуется проводить их по учебному пособию с использованием «Системы программирования на основе алгоритмического языка курса ОИВТ» (Е-практикум).

Пользователям КУВТ «Ямаха» Минпрос СССР передал пакет программ*, в котором есть все необходимое для проведения практикума в указанном виде. В школах с кабинетами, оснащенными ПЭВМ «Агат», практикум можно организовать на базе программной системы «Школьница». Программное обеспечение для школ, оснащенных КУВТ-86, необходимое для работы по данной программе, разрабатывается в НИИ информатики и вычислительной техники АПН СССР и в соответствии с планом будет поставляться централизованно к началу 1987/88 учебного года.

При планировании практических занятий наличие программного обеспечения должно учитываться в первую очередь. В связи с этим приведенная программа носит ориентировочный характер.

В порядке эксперимента по согласованию с местными органами народного образования практические занятия могут проводиться также в соответствии с программой конкурсного учебника (см.: Математика в школе. 1986. № 3; Микропроцессорные средства и системы. 1986. № 2; Информатика и образование. 1986. № 1).

В дальнейшем, по мере насыщения школ вычислительной техникой и соответствующим программным обеспечением, практические занятия по информатике будут в большей степени ориентироваться на формирование умений использовать готовое программное обеспечение.

* См.: Информатика и образование. 1987. № 1. Граница Е., Крюкова Т., Нестеров Н., Слуднов А. И-86.

ПОДДУБНАЯ Т.
Томский государственный университет

Информатика по опорным сигналам

В условиях безмашинного преподавания курса ОИВТ большую помощь учителю может оказать использовние опорных сигналов, введенных в педагогическую практику В. Ф. Шаталовым.

Мы начинаем публикацию методических рекомендаций по преподаванию курса с использованием пробного учебного пособия «Основы информатики и вычислительной техники» под редакцией А. П. Ершова и В. М. Монахова и листов опорных сигналов (ЛОС), разработанных Т. Н. Поддубной.

Публикуемые материалы были апробированы в школе. Использование ЛОС позволяет сократить время обучения, сформировать у учащихся не разрозненные понятия, а систему знаний и больше внимания обратить на выработку умений составления алгоритмов для ЭВМ.

«Жизнь»

> Бросая в воду камешки, смотри на круги, ими образуемые; иначе такое бросание будет пустой забавою.
> *Козьма Прутков*

Изучение курса информатики полезно начать с составления своего первого алгоритма. К нему предъявляется несколько требований: он должен быть интересен и несложен; хорошо было бы также, чтобы его можно было исполнить самому. Из множества удовлетворяющих этим требованиям алгоритмов выберем игру «Жизнь», придуманную сотрудником Кембриджского университета Дж. Конвеем.

«Жизнь» — многоклеточное сообщество, населяющее пустыни Флатландии. Для определенности предположим, что пустыня — прямоугольное поле, разделенное на квадратные клетки, каждая из которых может содержать организм — один элемент «жизни». Организмы умирают и рождаются, меняются поколения. Каждая клетка имеет 8 соседей (клетки, расположенные рядом с ней по горизонтали, вертикали и диагоналям).

M—1, K—1	M—1, K	M—1, K+1
M, K—1	M, K	M, K+1
M+1, K—1	M+1, K	M+1, K+1

Новое поколение организмов получается из предыдущего по простым правилам.

1. Если у некоторой клетки, в которой живет организм, меньше двух живых соседей (т. е. содержащих организм), то он погибает от одиночества. Если клетка имеет больше трех живых соседей, организм погибает от тесноты.

2. Если рядом с пустой клеткой оказывается ровно три клетки, в которых есть жизнь, то в ней рождается новый организм.

Итак, займемся игрой. Попробуем составить ее алгоритм при условии, что мы будем выполнять его сами.

Сначала весь алгоритм можно записать в виде только двух действий:

первое — занести в клетки пустыни исходную конфигурацию жизни, иными словами, в некоторые клетки поместить символ, которым мы будем изображать организм, например звездочку;

второе — построить новую конфигурацию в соответствии с заданными правилами получения нового поколения.

Рассмотрим эти два шага более подробно.

Для выполнения первого нужно найти клетки, в которые следует занести *. Заметим, что каждую клетку пустыни можно охарактеризовать двумя числами — номерами строки и столбца. Назовем их индексами клетки. Для человека совсем не трудно найти по индексам нужные клетки и поставить в них *, так как он видит и умеет считать. Для ЭВМ эта задача более сложна, но пока мы собираемся выполнять алгоритм сами.

Второй шаг уже не так прост, как первый; будет ли жизнь в какой-то клетке в следующем поколении, зависит от нескольких условий, а именно:

```
если в клетке нет жизни (т. е. *),
    то
        подсчитать число живых соседей
        клетки;
        если это число равно 3,
            то поставить в клетке * для
            следующего поколения
            иначе ничего не делать, т. е.
            оставить клетку пустой
    иначе (т. е. в клетке есть *)
            подсчитать число живых со-
            седей клетки;
    если это число равно 2 или 3
        то оставить в клетке * для сле-
        дующего поколения
        иначе убрать * из клетки, т. е.
        сделать ее пустой для следую-
        щего поколения.
```

Как теперь видно, второй шаг представляет собой одну инструкцию, но с возможностью разветвления. Обратите внимание на слова **если**, **то**, **иначе**.

Первое разветвление делается по условию наличия или отсутствия * в клетке. На каждом пути после этого разветвления стоят две инструкции — подсчет числа живых соседей (человеку это сделать несложно) и снова инструкция о разветвлении. Второе разветвление делается уже по другому условию — сколько живых соседей имеет рассматриваемая клетка.

Чтобы закончить рассмотрение второго шага, надо отметить, что написанную выше инструкцию следует выполнить для всех клеток пустыни, что можно записать так:

```
пока не просмотрели все клетки пустыни,
        выполнить следующие действия от
        начало до конец
начало
если в клетке нет *
    то подсчитать число живых со-
    седей клетки
        если это число равно 3
            то поставить в клетку *
            иначе ничего не делать
    иначе
            подсчитать число живых со-
            седей клетки
        если это число равно 2 или 3
            то оставить * в клетке
            иначе убрать * из клетки
конец
```

Очевидно, что действия от **начало** до **конец** идентичны для всех клеток и будут повторены столько раз, сколько клеток в пустыне.

Для проведения игры следует взять два поля (две «пустыни»). На одну наносится исходная конфигурация жизни. Для этого можно воспользоваться карточками с нарисованными на них *. На другом поле будет строиться новая конфигурация — следующее поколение. Можно построить несколько поколений, меняя «пустыни» местами на каждом шаге и предварительно очищая от * предыдущее поле (поле с предыдущим поколением).

Выводы и рекомендации

Игра «Жизнь» может быть рекомендована для первого занятия по курсу «Основы информатики и вычислительной техники». Она интересна — в этом ее главное достоинство.

Игра знакомит учащихся с понятием алгоритма как четкой последователь-

ности действий, ведущих к достижению некоторой цели.

Игра сразу вводит учащихся в класс базовых инструкций, которые они будут применять в дальнейшем для записи алгоритмов. Это — инструкции повторения (пока) и ветвления (если-то-иначе), а также инструкция последовательного выполнения действий в алгоритме.

В процессе игры учащиеся встречаются с различными типами величин. Символ * — литерная величина, а количество живых соседей — целая.

Игра демонстрирует учащимся метод последовательной разработки алгоритма (метод пошаговой детализации), с которым они ближе познакомятся впоследствии. Разработку алгоритма можно продолжить далее, расписывая более подробно, как подсчитать число живых соседей у клетки с индексами М и К.

На примере этого алгоритма можно обсудить вопрос о том, что должна уметь делать ЭВМ, чтобы быть его исполнителем (ориентироваться в двумерных таблицах, складывать целые числа, проверять условия, многократно повторять однотипные действия для разных значений и т. п.).

Пояснения к ЛОС 1

Приведенный лист опорных сигналов соответствует разделу «Первоначальные сведения об ЭВМ» пробного учебного пособия.

На схеме выделены основные блоки современной ЭВМ — память (ЗУ — запоминающее устройство), устройства ввода-вывода (УВВ) и центральный процессор, который для удобства проведения исторической аналогии представлен в виде двух блоков: арифметического устройства (АУ) и устройства управления (УУ).

В начале XIX в. английским инженером Чарльзом Бэббиджем была выдвинута идея создания счетной машины с программным управлением. Его ученица, дочь английского поэта Дж. Байрона леди Лавлейс (сейчас ее считают первой программисткой), писала, что машину Бэббиджа «нельзя смешивать с простыми счетными машинами. Она занимает совершенно особое место... среди механизмов, предоставляющих возможности комбинировать произвольные символы».

Функциональная схема машины Бэббиджа содержала блоки *склад*, *мельница* и *контора*. Они приведены на ЛОС и, думается, не нуждаются в особых пояснениях, как и факт их полного соответствия определенным функциональным блокам современной ЭВМ.

Память ЭВМ на листе опорных сигналов представлена в виде куба, разделенного на отдельные элементарные ячейки — места хранения единиц информации, обработкой которой занимается ЭВМ. Подобное деление, кроме того, что оно связано с понятием байта, а также более и менее крупных ячеек памяти (бит, слово и т. д.), позволяет просто объяснить один из основных принципов построения современных вычислительных машин, сформулированный Джоном фон Нейманом. Это принцип адресации ячеек памяти (опорный сигнал ПА).

Память ЭВМ, какую бы геометрическую форму она ни имела (куб, цилиндр, диск, лента), разбита на элементарные ячейки, которые пронумерованы подряд. Номер ячейки есть ее адрес. ЭВМ размещает и разыскивает информацию в памяти по адресам. Таким образом, каждая ячейка характеризуется своим адресом и содержимым (кодом, который в ней записан). Работа ЭВМ, собственно, и заключается в том, что она в соответствии с некоторой совокупностью команд меняет содержимое ячеек памяти.

Усвоить этот принцип помогает такой пример. Кто живет по адресу: ул. Ленина, д. 6, кв. 20? Сегодня один, завтра другой. Содержимое ячейки с данным адресом может быть разным в разное время.

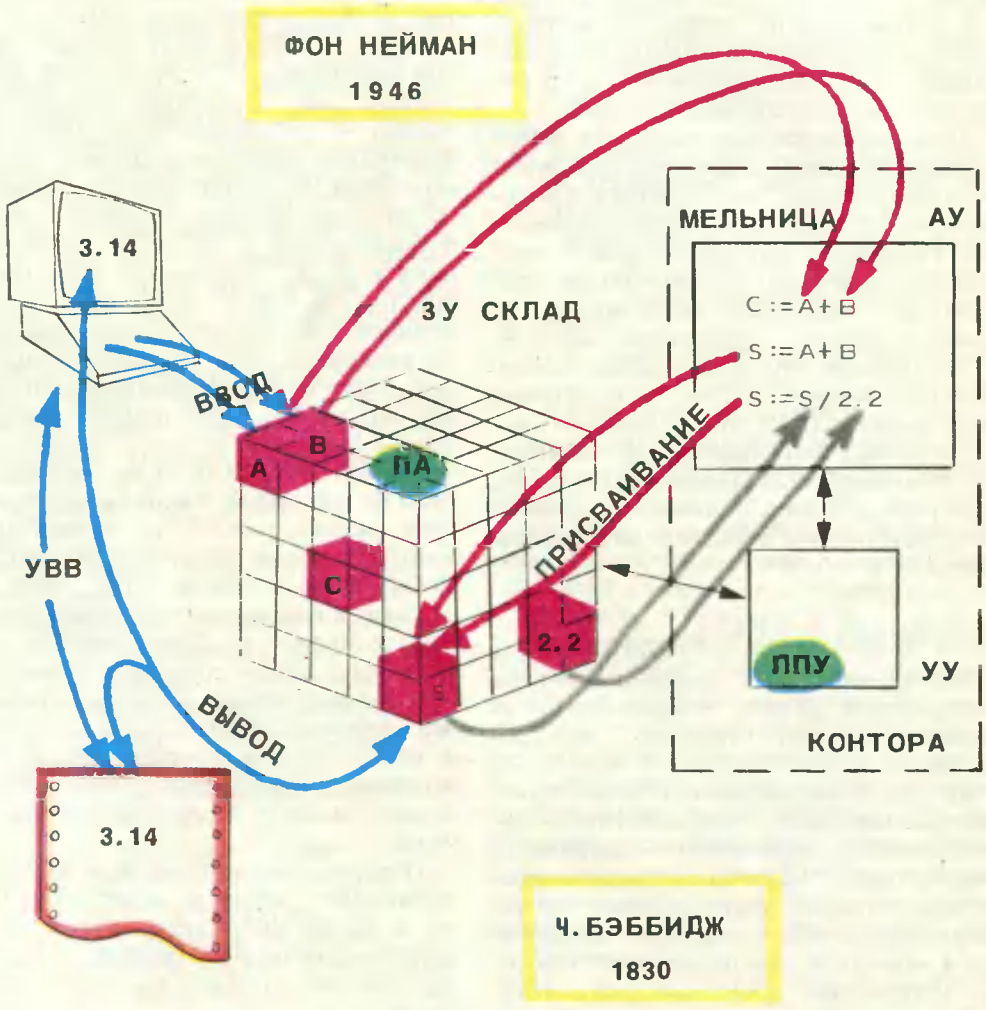
Первоначально информация заносится в память ЭВМ с помощью специальных команд (операторов) ввода, и, хотя в учебном пособии их нет, упомянуть их полезно. Команды ввода обеспечивают ввод информации в оперативную (внутреннюю) память ЭВМ с различных внешних носителей и устройств — перфокарт, магнитных лент и магнитных

ЛОС 1

Ибо это недостойно совершенства человеческого, подобно рабам тратить часы на вычисления.

Готфрид В. Лейбниц

Бэббидж говорил, что он заставляет машину bite its own tail (кусать себя за хвост).



дисков, с клавиатур терминалов. На ЛОС этот момент изображен стрелкой, в качестве внешнего устройства показана клавиатура персональной ЭВМ.

Подобное, в некоторой степени углубленное, представление структуры памяти ЭВМ является важным в методическом плане, так как обучение основам алгоритмизации должно строиться все-таки в расчете на то, что исполнителем алгоритма будет ЭВМ. А это значит, что первичные знания о ней необходимы. Например, при изучении раздела «Алгоритмы работы с величинами» важно показать, что отнесение величин к различным типам (цел, вещ, нат, лит) обусловлено, в частности, и тем, что они имеют разное внутреннее (в памяти

ЭВМ) представление, в частности могут иметь разную длину.

Кроме того, знание структуры памяти позволяет уже при изучении этого материала основательно рассмотреть механизм выполнения алгоритма на ЭВМ — второй основной принцип, сформулированный фон Нейманом, — принцип программного управления (опорный сигнал **ПУ**). На ЛОС в квадрате, соответствующем арифметическому устройству, помещена последовательность трех команд — три оператора (так они называются в алгоритмических языках) присваивания значений переменным C и S. При записи операторов использован знак операции присваивания :=. После выполнения операторов C:=A+B и S:=A+B переменные C и S получат значение, равное сумме значений переменных A и B, которые были введены в память ЭВМ с помощью команд ввода. После выполнения оператора S:=S/2.2 переменная S получит новое значение, которое равно старому значению этой переменной, разделенному на константу 2.2. Последняя тоже берется из некоторой ячейки памяти ЭВМ.

Принцип программного управления — второй основной принцип современных вычислительных машин. Он заключается в том, что ЭВМ работает сама, без участия человека, по программе (последовательности команд), которая находится в ее памяти, т. е. предварительно вводится в нее с внешнего устройства. ЭВМ выбирает команды из памяти одну за другой, анализирует их в арифметическом устройстве, выполняет и результаты снова возвращает в память. Всем этим процессом управляет устройство управления. Именно такой способ работы имел в виду Ч. Беббидж, когда сказал, что его машина «кусает сама себя за хвост».

Остановимся еще на одном моменте. Команды присваивания S:=A+B и S:=S/2.2 находились в памяти ЭВМ и были выбраны в арифметическое устройство последовательно друг за другом. После выполнения первой команды в ячейку памяти для переменной S занеслось значение, равное сумме значений переменных A и B, а после выполнения второй — в 2,2 раза меньше. Второй оператор присваивания требует особого внимания при объяснении, так как при его выполнении переменная, получающая значение и стоящая слева от знака присваивания, и переменная, значение которой используется при вычислении выражения, записанного справа от знака присваивания, находятся в одной и той же ячейке памяти.

Таким образом, вынесенные на ЛОС команды присваивания помогают сделать два важных дела: пояснить принцип программного управления и смысл важнейшей операции — операции присваивания. Она легко воспринимается на этом этапе, так как понятие переменной, выражения и значения выражения хорошо знакомы учащимся из курса алгебры.

И еще одно замечание.

На ЛОС 1 выделены два вида операций (команд), которые позволяют менять содержимое ячеек памяти ЭВМ, — ввод и присваивание.

В чем их разница?

При вводе информация поступает с внешнего устройства (извне).

При присваивании информация (новое значение) создается при вычислении выражений внутри ЭВМ.

Опорные сигналы, связанные с выводом информации, не требуют дополнительных комментариев.

Отметим, что фамилии Ч. Бэббидж и Дж. фон Нейман, а также соответствующие даты несут в себе важный исторический аспект развития информатики. Они могут быть основой для интересной внеклассной и кружковой работы учащихся. Биография Ч. Бэббиджа и его ученицы Ады Лавлейс (см., например: *Апокин И. А., Майстров Л. Е., Эдлин И. С.* Чарльз Бэббидж. М.: Наука, 1981) — тема для нескольких занятий, которые будут способствовать более глубокому усвоению материала.

Выводы и рекомендации

Информация на ЛОС 1 — это систематизированное наглядное представление следующих понятий.

Функциональная схема ЭВМ.

Структура оперативной памяти ЭВМ.

Принцип адресации ячеек памяти ЭВМ.

Принцип программного управления ЭВМ.

Операция ввода информации и операция присваивания, их роль в алгоритмах.

На ЛОС при желании можно дополнительно вынести информацию о количественных характеристиках современных вычислительных машин, таких, как объем памяти (внешней и оперативной) и быстродействие, а также их нарастание со временем.

Пояснения к ЛОС 2

К моменту изучения информации, помещенной на ЛОС 2 (§ 2 учебного пособия), учащиеся уже должны хорошо освоить понятие алгоритма. И хотя оно дается им в интуитивном определении, этого достаточно (как, между прочим, и почти всем программистам-практикам) для того, чтобы грамотно записывать алгоритм как последовательность команд (инструкций) для достижения некоторой цели, например решения или моделирования поставленной задачи на ЭВМ.

При рассмотрении ЛОС 1 были обсуждены две важные команды, без которых нельзя обойтись при записи алгоритмов:

ввод значений переменных;

присваивание значений переменным.

На ЛОС 2 обсуждается вопрос о порядке выполнения команд, с помощью которых записан алгоритм. Этот порядок (его методологическая суть) хорошо может быть прослежен на примере алгоритма игры «Жизнь».

1. Команды выполняются последовательно друг за другом.

2. В алгоритме может быть некоторое условие, определяющее порядок выполнения команд, — ветвление на две последовательности, из которых в алгоритме выполняется только одна, в зависимости от выполнения или невыполнения этого условия.

3. В алгоритме может быть предусмотрена возможность многократного повторения определенной последовательности команд, если она идентично выполняется для нескольких различных значений каких-то величин.

На ЛОС 2 эти три вида инструкций, управляющих порядком выполнения команд алгоритма, ассоциативно связаны с опорным сигналом в виде светофора.

Светофор — прибор, управляющий движением. Что характерно для него?

1. Цветовые сигналы загораются последовательно.

2. Цвет сигнала определяет последовательность действий движущегося объекта в виде ветвления. Например, в упрощенном варианте для пешехода:

```
если цвет зеленый
    то Иди!
    иначе
    если цвет желтый
        то Жди!
        иначе Стой!
```

3. Работа светофора — это многократное повторение смены трех цветов от начала движения транспорта и пешеходов до его окончания.

Кроме служебных слов, применяющихся в учебном пособии для записи команд ветвления и повторения, на ЛОС приведены служебные слова, применяющиеся для записи этих команд в современных алгоритмических языках:

IF — если
THEN — то
ELSE — иначе

WHILE — пока
DO — делай
BEGIN — начало (нц)
END — конец (кц)

На ЛОС приведены также схематические изображения управляющих команд с использованием стандартной символики.

Одна или несколько команд, которые в алгоритме рассматриваются как одно целое и называются серией, изображаются в прямоугольнике с одним входом и одним выходом; условие — в ромбе с одним входом и двумя выходами, над которыми написаны слова «да» и «нет», означающие выполнение и невыполнение условия соответственно.

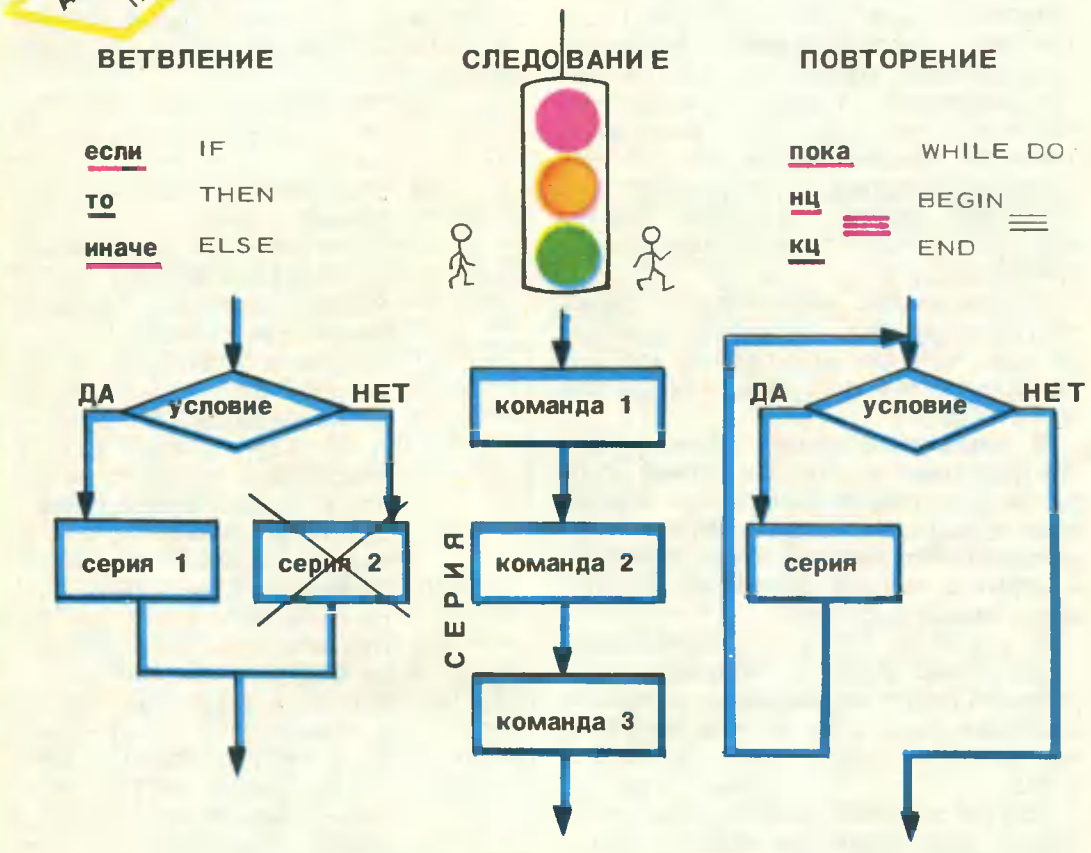
Пути выполнения алгоритма указываются прямыми линиями со стрелками.

Обратите внимание: на ЛОС неявно

Что вы привыкли делать дома
Единым махом, наугад,
Как люди пьют или едят,
Вам расчленят на три приема.
Иоганн В. Гёте

АЛЬ ХОРЕЗМИ IX век

УПРАВЛЕНИЕ В АЛГОРИТМЕ



определено, что серия — это последовательность любых команд, выполняемых друг за другом. Иначе говоря, внутри серии могут быть не только простые команды (например, присваивания и ввода), но и команды ветвления и повторения. Таким образом, правильно будет при записи алгоритма в случае необходимости внутри одной команды ветвления (хоть после то, хоть после иначе) писать другие команды ветвления или команды повторения. Аналогично внутри команды повторения могут быть другие команды повторения (но только целиком!) и команды ветвления. Следовательно, приведенные определения управляющих команд позволяют записывать сколь угодно сложные алгоритмы.

Зачеркнутый на ЛОС прямоугольник «серия 2» означает, что если условие предполагает наличие действий (последовательности команд) только в одном случае (либо при «да», либо при «нет»), а в альтернативном варианте предполагается отсутствие действий (пустое

действие, или ничегонеделание), то можно пользоваться неполной формой команды ветвления:

если условие
то серия
все

(в записи отсутствует иначе).

Выводы и рекомендации

На ЛОС 2 образно представлена информация о системе базовых управляющих команд (или операторов), достаточных для записи любого алгоритма. При изучении конкретных языков программирования учащиеся неминуемо встретятся и с другими командами; важность приведенных в том, что они формируют культуру алгоритмического мышления, определяют главное свойство алгоритмов — простоту их написания и понимания.

Использование английских служебных слов на ЛОС облегчает усвоение их при изучении конкретных алгоритмических языков, таких, как Бейсик или Паскаль.

В заключение этого раздела рассмотрим одну задачу, при записи алгоритма для которой необходимо использование вложенных друг в друга команд повторения (примеров с вложенными командами ветвления достаточно в пробном учебном пособии).

Условие задачи. Имеется некоторая сумма денег А, например один рубль. Требуется написать алгоритм, дающий все варианты размена этой суммы монетами достоинством 15, 10, 5 и 1 коп.

Для сокращения записи введем следующие обозначения для величин, которые нам понадобятся:

К15, К10, К5, К1 — количества монет достоинством 15, 10, 5 и 1 коп. соответственно в конкретном варианте размена.

ОСТ15, ОСТ10, ОСТ5 — остаток от деления некоторого целого числа на 15, 10 и 5 соответственно.

С15, С10, С5 — целая часть, получающаяся при делении некоторого целого числа на 15, 10 и 5 соответственно.

Напишем сначала алгоритм для получения одного варианта размена. Его команды будем нумеровать целыми числами 10, 20, 30 и т. д. Это удобно, во-первых, потому, что можно не переписывать номера многократно в процессе разработки алгоритма, и, во-вторых, потому, что есть возможность вставить необходимую команду в уже имеющуюся последовательность команд, не нарушая при этом их порядок выполнения. Например, между 20-й и 30-й командами можно вставить команды с номерами 21, 22 и т. д. до 29. Ниже приведен алгоритм одного варианта размена с необходимыми комментариями в круглых скобках.

```
10  Вычислить целую часть С15 от деления заданной суммы А на 15
20  Присвоить величине К15 значение величины С15 (это будет максимальное значение пятнадцатикопеечных монет, которое может быть использовано при размене А)
30  Получить остаток ОСТ15 от деления целого числа А на 15 по формуле ОСТ15=А—15×К15 (число ОСТ15 теперь надо разделить на 10, чтобы узнать, сколько будет десятикопеечных монет в данном варианте размена)
40  Вычислить целую часть С10 от деления ОСТ15 на 10
50  Присвоить значение величины С10 величине К10
60  Получить остаток ОСТ10 от деления ОСТ15 на 10 по формуле ОСТ10=ОСТ15—10×К10 (число ОСТ10 надо разделить на 5, чтобы узнать, сколько будет монет по 5 коп. в данном варианте размена)
70  Вычислить целую часть С5 от деления ОСТ10 на 5
80  Присвоить значение величины С5 величине К5
90  Получить остаток ОСТ5 от деления ОСТ10 на 5 (это будет количество монет по 1 коп. в данном варианте размена)
100 Присвоить величине К1 значение величины ОСТ5
110 Напечатать значения величин К15, К10, К5, К1
```

Таким образом, составленный нами алгоритм получения одного варианта размена представляет собой серию из одиннадцати команд.

Получить следующий вариант размена можно, например, так: уменьшим на единицу количество монет по 5 коп. и разменяем этот пятачок копеечными монетами. Таким образом, в этом варианте не меняется количество монет по 15 и 10 коп., количество монет по 5 коп. уменьшается на единицу, а количество монет по 1 коп. увеличивается на 5.

Аналогично можно поступить еще несколько раз, до тех пор, пока число пятаков не станет равно нулю. Затем можно, не изменяя числа монет по 15 коп., построить все варианты размена, при которых меняются количества монет достоинством 10, 5 и 1 коп. Сам алгоритм размена не изменяется. Таким образом, мы имеем дело с многократным повторением идентичных действий на разных уровнях (не меняется число монет по 15 и 10 коп. — меняется число монет по 5 и 1 коп.; не меняется число монет по 15 коп. — меняется число монет по 10, 5, 1 коп.). И наконец, еще одну группу вариантов можно получить, уменьшая последовательно на единицу число монет по 15 коп.

Проследим сказанное на приведенной выше последовательности операторов для получения одного варианта размена.

Команды с номерами 90, 100 и 110 должны выполняться многократно для различных значений величины K5; изменение ее значения должно происходить после команды с номером 110. Добавим к алгоритму требующуюся команду повторения, присвоив ей соответствующий номер, а также команду изменения K5 на —1 на каждом шаге команды повторения. Ниже схематически изображен видоизмененный фрагмент алгоритма:

```
85  пока K5⩾0
        нц
          90 .....
          100 .....
          110 .....
          115 K5:=K5—1
    кц
```

Следующие варианты размена получаются при изменении числа монет по 10 коп. от максимально возможного до нуля при фиксированном числе монет по 15 коп. Следовательно, можно оформить аналогичную команду повторения:

```
55  пока K10⩾0
        нц
          60 .....
          70 .....
          80 .....
          85 пока K5⩾0
                нц
                  90 .....
                  100 .....
                  115   K5:=K5—1
                кц
                  120   K10:=K10—1
        кц
```

И, наконец, оформим последнюю команду повторения для изменения количества монет по 15 коп. и получим окончательный текст алгоритма. Места вставки команд повторения указаны на исходном тексте алгоритма стрелочками.

```
Ввести заданное число A
Вычислить целую часть C15 от деления
A на 15
K15:=C15
пока K15⩾0
  нц OCT15:=A—15×K15
  Вычислить целую часть C10 от деления
  OCT15 на 10
  K10:=C10
    пока K10⩾0
        нц  OCT10:=OCT15—10×K10
  Вычислить целую часть от деления
  OCT10 на 5
  K5:=C5
    пока K5⩾0
        нц OCT5:=OCT10—5×K5
           K1:=OCT5
  Напечатать значения величин K15,
  K10, K5, K1
  K5:=K5—1
        кц
          K10:=K10—1
    кц
      K15:=K15—1
кц
```

Является ли данный алгоритм самым лучшим?

С уверенностью можно сказать лишь одно — он является достаточно простым, что соответствует одному из требований психологии программирования. Про-

граммируй сначала понятно, а потом оптимально. Следующим этапом работы может быть его улучшение, в итоге можно прийти к совсем другому алгоритму. Для того чтобы в алгоритме команда повторения работала правильно — именно столько раз, сколько нужно, и каждый раз для новых значений каких-то конкретных величин, ее оформление требует соблюдения двух правил.

1. Условие окончания работы команды повторения (цикла) содержится в ее заголовке. В нашем примере — это условия K15≥0, K10≥0, K5≥0. Программист должен не забывать изменить значение переменной, определяющей конец работы оператора цикла, в теле цикла (последовательности повторяющихся команд). В нашем примере это команды K15:=K15—1; K10:=K10—1; K5:==K5—1. Обычно подобные команды пишутся в самом конце последовательности команд, составляющих тело цикла, т. е. перед служебным словом кц.

2. В заголовке и теле цикла часто используются некоторые величины с заранее известными значениями, например для проверки условий или для получения значений новых величин. Программист должен не забывать присвоить им значение до входа в цикл. Команды, применяющиеся для этого, обычно называют командами инициализации цикла. В нашем примере команды инициализации должны быть обязательно написаны для величин K15, K10, K5, которые используются уже в заголовке цикла.

Таким образом, команда повторения с точки зрения дисциплины программирования может быть схематически представлена так:

Операторы инициализации цикла
Заголовок цикла (пока)
нц
тело цикла
операторы, определяющие конец работы цикла
кц

Пояснения к ЛОС 3

Алгоритмы пишутся для обработки некоторой информации. Обычно ее называют данными или величинами. В дальнейшем мы будем использовать эти слова как синонимы. После рассмотренных примеров уже должно быть ясно, что данные могут быть различными как по типу, так и по способу своей организации.

Классификации данных и описанию их характеристик посвящен ЛОС 3 (§ 3, разделы 5, 10 пробного учебного пособия).

Для классификации данных используется опорный сигнал в виде кубика. На его гранях представлены характеристики данных по двум направлениям.

Передняя грань делит данные на четыре класса — вещественные, целые, натуральные и литерные. Служебные слова, описывающие соответствующий тип, приведены в правом нижнем углу ЛОС. Почему в ЭВМ применяется подобное деление?

1. Данные разного типа имеют разное внутреннее представление в ЭВМ. Например, данное литерного типа занимает в памяти столько байтов, сколько содержит символов, так как один символ размещается в одном байте. Данные типа вещ почти всегда занимают 4 байта (32 двоичных разряда), а данные целого типа (цел) — 2 байта (16 двоичных разрядов).

2. Для данных разного типа в ЭВМ могут быть предусмотрены разные операции. Например, для данных типа цел очень часто вводятся дополнительные арифметические операции целочисленного деления и вычисления остатка от деления.

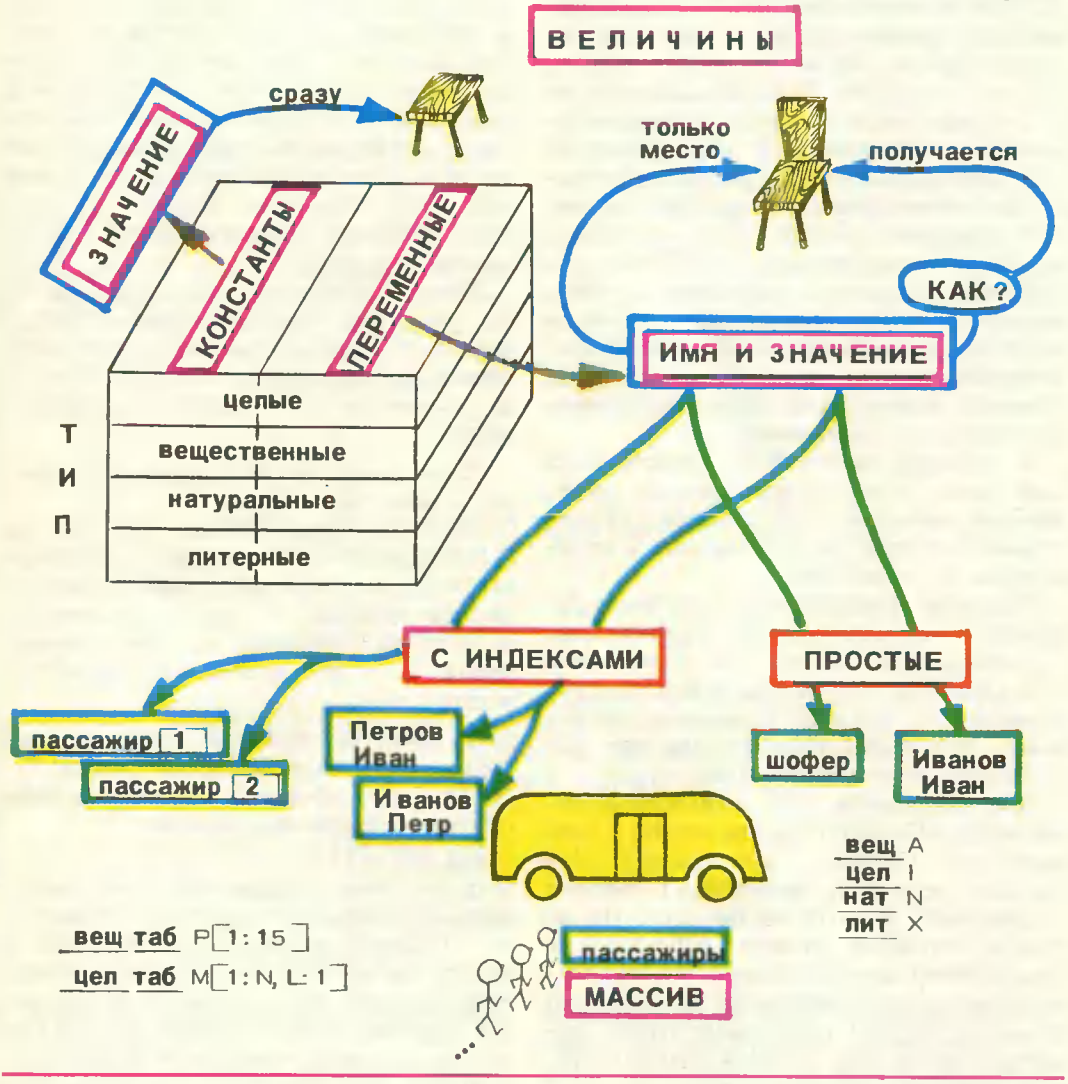
На верхней грани кубика — другая линия классификации. В процессе работы алгоритма данные могут оставаться постоянными. Это — константы. Но они могут и изменять свое значение. Это — переменные. Подобное деление является очень важным, так как позволяет остановиться на фундаментальном понятии программирования — имени переменной (в алгоритмических языках как синоним употребляется слово «идентификатор»), а также на механизме распределения памяти под величины, использующиеся в алгоритме.

ЭВМ, читая алгоритм, для всех величин отводит место в памяти. На ЛОС этот факт связан с табуретом и стулом. Что же разного в процедуре отведения

ЛОС 3

Целые числа сотворил господь бог, все остальное — дело рук человеческих.
Леопольд Кронекер

Здесь у меня столовая.
Вся мебель в ней дубовая.
Вот это стул — на нем сидят,
Вот это стол — за ним едят.
С. Маршак



места под константы и переменные?

Константа однозначно, раз и навсегда, задается своим значением. 3.14 — это только 3.14, и более ничего. Отводя ей место в памяти, ЭВМ сразу помещает туда константу и запоминает (для себя) адрес этого места. Потом в процессе работы алгоритма она будет находить константу по ее адресу, который (подчеркнем это!) остается неизвестным программисту.

Переменная может принимать разные значения в процессе работы алгоритма. Поэтому для нее и вводится обозна-

чение — имя. Когда ЭВМ встречает в алгоритме произвольное имя, она отводит ему только место в памяти (опорный сигнал — «стул»). Значение же (то, что будет находиться на стуле) получается в различных командах алгоритма. Подчеркнем, что место отводится при первой встрече имени, т. е. один раз, значение же может меняться много раз. На ЛОС стоит вопрос «как?». Как можно менять значение одной и той же переменной? Эта задача уже обсуждалась при рассмотрении ЛОС 1. Существуют две возможности — команда ввода и команда присваивания. В первом случае значение переменной берется с внешнего носителя. Во втором случае оно может вычисляться по достаточно сложному выражению процессором ЭВМ.

Чтобы закончить описание кубика, отметим, что он подчеркивает возможность использования в алгоритмах как констант целого, вещественного, натурального и литерного типа, так и соответствующих переменных.

В нижней части ЛОС представлена еще одна линия классификации переменных величин — по способу организации. Это простые переменные и переменные с индексами.

Простые переменные в программировании называют иногда скалярными.

Переменные с индексами — элементы совокупностей однотипных величин, называемых в программировании массивами. В учебном пособии для них использован термин «таблицы».

Для пояснения сути этой классификации на ЛОС использован сигнал в виде автобуса. Шофер — это простая, скалярная, величина, являющаяся именем переменной. В автобусе он один. Но на одном автобусе может работать не один шофер. Следовательно, это имя может иметь разные значения. Для данного примера такими значениями будут фамилия, имя и отчество каждого шофера, т. е. значения литерного типа.

Автобус перевозит пассажиров. Они образуют массив переменных. Общее имя этого массива — ПАССАЖИР. В автобусе каждый пассажир полностью определяется номером места, которое он занимает (прочие качества пассажира здесь не имеют значения). Таким образом, к каждому из них можно обратиться следующим образом:

ПАССАЖИР [1],
ПАССАЖИР [2]

и т. д. Это и есть имена переменных с индексами.

Достаточно указать номер места (индекс), как нужный пассажир отзовется и можно узнать его значение — имя, фамилию и отчество, т. е., как и в случае с шофером, значение литерного типа. Эта аналогия позволяет зафиксировать внимание на сущности понятия имени переменной, так как в самой аналогии слово «имя» используется дважды — как понятие программирования и как значение литерного типа. Различение этих двух моментов свидетельствует о понимании сути дела.

Таким образом, массивы (таблицы) — это совокупности однотипных переменных, в которых каждый элемент однозначно определяется своим местом в ряду элементов, образующих эту совокупность.

Слева внизу на ЛОС приведены примеры описания таблиц.

Рассмотренный только что массив ПАССАЖИР — линейный (одномерный). Можно себе представить и многомерные массивы — двух-, трехмерные и т. д. Их элементы задаются двумя, тремя и т. д. индексами соответственно.

В качестве примера использования двухмерного массива рассмотрим пустыню (см. игру «Жизнь»). Она может быть описана следующим образом:

лит таб Р[1:10, 1:20]

Это описание определяет поле, разделенное на элементы (клетки), состоящее из 10 строк и 20 столбцов (естественно, число строк и столбцов может быть любым). Всех клеток, т. е. элементов массива, будет $10 \times 20 = 200$. В каждом из них может находиться символ *, а может не быть ничего (в таком случае говорят, что в данном элементе массива присутствует символ ␣ (пробел)).

Рассмотрим фрагмент алгоритма для задачи «Жизнь», где требуется подсчитать число соседей, содержащих *, для заданной клетки.

Клетка как элемент двумерного мас-

сива определяется парой индексов. Пусть имена индексов будут М и К. Мы специально выбираем их не константами, так как они изменяются, обеспечивая просмотр всех клеток пустыни. Ниже изображены все восемь соседей клетки с индексами М и К; для подсчета «живых» соседей надо просмотреть их.

М—1, К—1	М—1, К	М—1, К+1
М, К—1	М, К	М, К+1
М+1, К—1	М+1, К	М+1, К+1

Заметим, что индекс строки при просмотре изменяется от М—1 до М+1, а индекс столбца — от К—1 до К+1, но элемент с индексами М и К просматривать не надо. Очевидно, что мы будем производить идентичные действия для каждой из восьми клеток и, следовательно, данный фрагмент алгоритма может быть оформлен как вложенные команды повторения.

ЧИСЛО ЖИВЫХ СОСЕДЕЙ — имя переменной, которая используется для получения искомой величины.

В круглых скобках — комментарии.

Выводы и рекомендации

ЛОС 3 представляет собой образное и систематизированное изображение информации о величинах. Он дает знания как отдельных характеристик величин, так и их взаимодействия при различных видах организации.

Важным и существенным моментом, выделенным на ЛОС, является вопрос о предоставлении места в памяти разным величинам. Простым переменным и константам — одно место, массивам — столько мест, сколько элементов в массиве, хотя массив имеет одно общее имя.

Переменные с индексами не описываются в алгоритме каждая в отдельности. Описывается сразу весь массив.

```
СТРОКА:=М—1; ЧИСЛО ЖИВЫХ СОСЕДЕЙ:=0;
пока СТРОКА≤М+1
    нц
      СТОЛБЕЦ:=К—1;
      пока СТОЛБЕЦ≤К+1
          нц
            если не (СТРОКА=М И СТОЛБЕЦ=К)
            то если Р [СТРОКА, СТОЛБЕЦ]=«*»
                  то ЧИСЛО ЖИВЫХ СОСЕДЕЙ:=
                  ЧИСЛО ЖИВЫХ СОСЕДЕЙ+1
                  все
            все;
            СТОЛБЕЦ:=СТОЛБЕЦ+1
      кц; (по столбцам)
    СТРОКА:=СТРОКА+1
кц; (по строкам)
```

Здесь использованы следующие имена переменных:

СТРОКА — имя переменной целого типа, значение которой изменяется при работе фрагмента алгоритма от М—1 до М+1;

СТОЛБЕЦ — имя переменной целого типа, значение которой изменяется от К—1 до К+1;

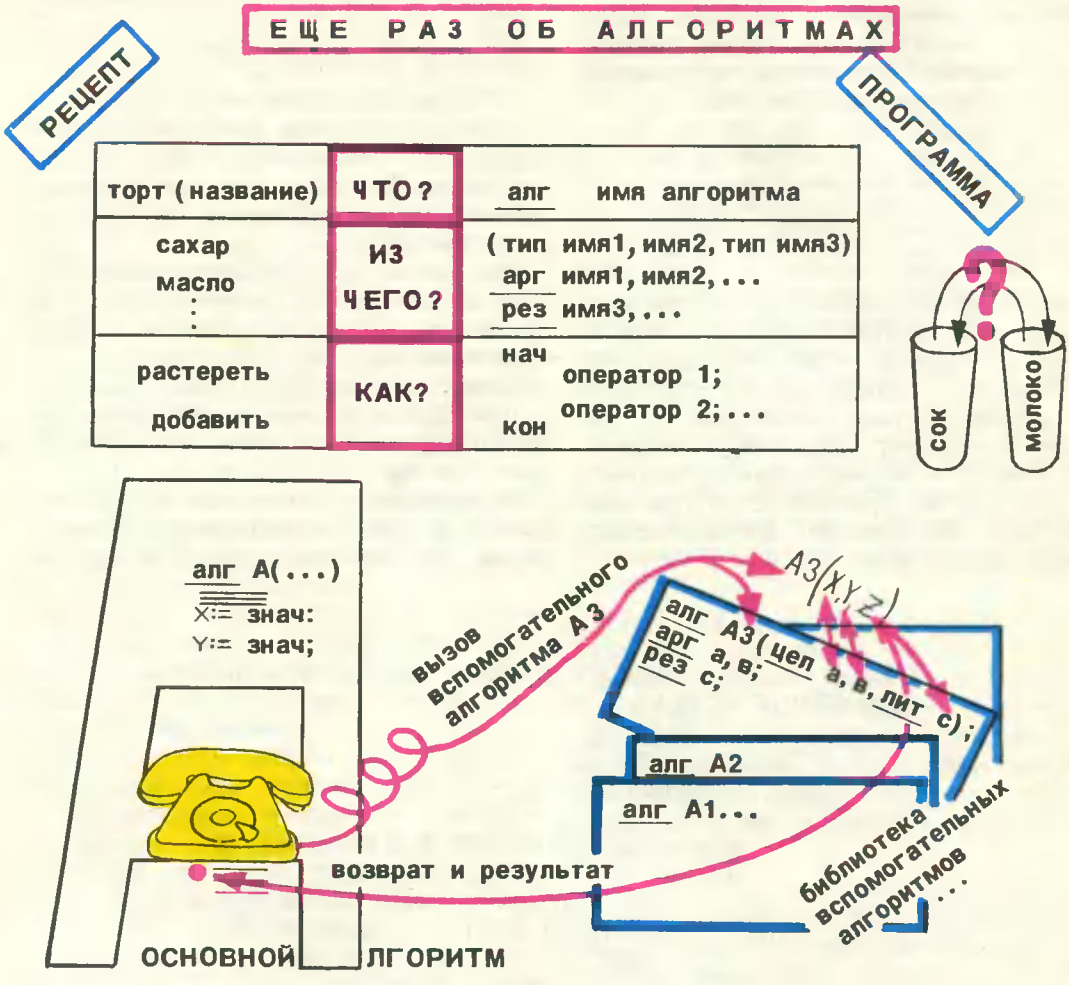
Р — имя массива клеток, представляющего пустыню;

В описании массива (таблицы) указывается его имя, тип его элементов и границы изменения индексов (иными словами, число элементов массива).

Пояснения к ЛОС 4

ЛОС 4 снова возвращает нас к понятию алгоритма. Но этот возврат происходит уже на более высоком уровне знания. Теперь можно взглянуть на это

...я — части часть,
которая была когда-то всем.
Иоганн В. Гёте



понятие с другой точки зрения — конструктивной. Почему именно так, а не иначе выбран способ записи алгоритма? Чем это обусловлено?

Житейским примером, который может быть использован для пояснения способа записи алгоритма, является любой кулинарный рецепт — *как* приготовить определенное блюдо (*что*) из определенных продуктов (*из чего*)? Аналогия полная. Имя алгоритма — это название производимого продукта. Если алгоритм пишется для ЭВМ (т. е. составляется программа на определенном алгоритмическом языке), то его зачастую так и называют — программный продукт. Указание типов имен, являющихся аргументами (входами) и результатами (выходами) алгоритма — это задание конкретных характеристик величин, для обработки которых предназначен алгоритм. И, наконец, последовательность команд (операторов) от **нач** до **кон** — это последовательность действий, необходимых для получения результата.

На ЛОС помещен опорный сигнал в виде двух стаканов с разными жидкостями. Он связан с понятием промежуточной величины (§ 3, раздел 7). Можно ли поменять местами содержи-

мое двух стаканов, не пользуясь третьим, промежуточным? Нет, если речь идет не о фокусах. Так и при исполнении любого алгоритма могут понадобиться некоторые вспомогательные величины. Например, вспомогательной (промежуточной) переменной является переменная D (дискриминант квадратного уравнения) при написании алгоритма нахождения корней квадратного уравнения. Если в алгоритме требуется поменять друг с другом значения двух переменных, например A и B (как в случае жидкостей в двух стаканах), то в нем также не обойтись без промежуточной переменной C; смена значений может быть реализована с помощью трех операторов присваивания:

C:=A; A:=B; B:=C.

Промежуточные величины не являются ни аргументами, ни результатами, но в теле алгоритма (от нач до кон) они должны быть обязательно описаны, так как им должно быть предоставлено место в памяти ЭВМ в соответствии с их типом.

Если в последнем примере в A и B содержались литерные значения, то и промежуточная переменная C должна быть описана как переменная литерного типа. Если же A и B были целыми числами, то переменная C должна иметь целый тип.

В реальных программах количество промежуточных величин может значительно превышать количество аргументов и результатов.

Вторая часть ЛОС 4 посвящена вопросам оптимального использования труда программистов. Дело в том, что хороший алгоритм, составленный однажды, не имеет тенденции к порче и потому им можно пользоваться как угодно долго, а чтобы не переписывать каждый раз, из хороших алгоритмов составляют библиотеки алгоритмов и определяют общие правила пользования такими библиотеками. Все алгоритмы, занесенные туда, называются вспомогательными (§ 4, раздел 11). В алгоритмических языках они называются процедурами и функциями. Для того чтобы воспользоваться вспомогательным алгоритмом при написании другого алгоритма (на ЛОС последний помещен в большую букву A и назван основным; его имя — тоже A), необходимо в нужном месте написать команду (оператор) вызова вспомогательного алгоритма. Она имеет следующий формат:

ИМЯ вспом. алг. (список конкретных имен, для которых должен проработать вспомогательный алгоритм)

Это показано на ЛОС 4. К моменту вызова (опорный сигнал в виде телефона) вспомогательного алгоритма с именем A3 в основном алгоритме должны быть обязательно описаны и определены значения переменных, которые при вызове будут переданы как аргументы. Также должна быть описана величина, которая будет передана во вспомогательный алгоритм для получения результата его работы.

Работа с вспомогательными алгоритмами требует от программиста внимательности и аккуратности; все величины, передаваемые из основного алгоритма во вспомогательный, должны соответствовать «заказанному» типу. Сопоставляются они по порядку следования, что отражено на ЛОС стрелочками.

Алгоритм, вспомогательный в одном случае, может оказаться основным по отношению к другому алгоритму, если во время своей работы он обращается к нему. Этот другой алгоритм будет по отношению к вызывающему вспомогательным.

На ЛОС 4 стрелочкой с надписью «возврат и результат» подчеркнут тот факт, что возвращение результата работы вспомогательного алгоритма происходит всегда в вызывающий, т. е. в основной для него, алгоритм к команде, следующей за командой вызова.

Оформим в виде вспомогательного алгоритм подсчета числа «живых» соседей для клетки с индексами M и K.

```
алг СЧЕТ (цел L, N, лит таб P[1:L, 1:N], цел M, K цел KS);
арг L, N, R, M, K
рез KS
НАЧ цел СТРОКА, СТОЛБЕЦ;
```

```
СТРОКА:=М—1; KS:=0;
пока СТРОКА⩽М+1
    нц
      СТОЛБЕЦ:=К—1;
      пока СТОЛБЕЦ⩽К+1
          нц
            если не (СТРОКА=М и СТОЛБЕЦ=К).
              то если Р [СТРОКА, СТОЛБЕЦ]=«*»
                    то KS:=KS+1
                 все;
            все;
            СТОЛБЕЦ:=СТОЛБЕЦ+1;
          кц; (по столбцам)
          СТРОКА:=СТРОКА+1;
        кц; (по строкам)
кон (алгоритма СЧЕТ)
```


Теперь в алгоритме для задачи «Жизнь» везде, где стоит команда «подсчитать число живых соседей клетки», надо написать вызов вспомогательного алгоритма СЧЕТ:

СЧЕТ (10, 20, Р, М, К, ЧИСЛО ЖИВЫХ СОСЕДЕЙ)

Выводы и рекомендации

ЛОС 4 является своего рода завершением изучения теоретического материала, связанного с правилами написания и оформления алгоритмов.

При изучении ЛОС 4 и соответствующего материала по учебному пособию следует обратить внимание на такой важный момент.

Если алгоритм предназначен для работы в качестве основного, т. е. к нему не предполагается обращение в дальнейшем, то первым оператором в его теле должен быть оператор ввода необходимых для работы значений величин. В учебном пособии об операторах ввода речи не ведется, но, если предполагается хоть какое-то общение с ЭВМ, о вводе говорить просто необходимо.

Можно пойти по другому пути — все алгоритмы оформлять как вспомогательные. Тогда ввода в них не требуется, так как конкретные значения аргументов (входных величин) передаются им при вызове. Именно этот путь и выбран в учебном пособии.

Мы не привели полный текст алгоритма игры «Жизнь», но все необходимое для него подготовлено. Попробуйте скомпоновать этот алгоритм и проверить его правильность, поставив себя на место ЭВМ, т. е. выполнить его хотя бы для трех шагов самого внешнего цикла.

Опять скажу: никто не обнимет необъятного!

Козьма Прутков

Нам говорят «безумец» и «фантаст»,
Но, выйдя из зависимости грустной,
С годами мозг мыслителя искусный
Мыслителя искусственно создаст.

Иоганн В. Гёте

БУЛАНОВА Н., ВОЛКОВ Д., ШИРКОВ П.

Графика на алфавитном дисплее

Эта статья посвящена тому, как применять компьютерные средства для создания изображения в курсе «Основы информатики и вычислительной техники»; описаны особенности алфавитно-цифровых дисплеев и АЦПУ как графических устройств вывода. Приводятся задачи с краткими методическими указаниями по их решению. Решения даются на языках программирования Фортран и Бейсик.

В основу статьи положен опыт работы со старшеклассниками общеобразовательных школ Брежневского района Москвы.

Курс «Основы информатики и вычислительной техники» не должен быть оторван от остальных школьных предметов. Одна из его задач — обобщать и систематизировать знания учащихся об использовании ЭВМ, полученные из других школьных предметов, и формировать научное мышление, основы которого закладываются в наиболее раннем возрасте. Только такой подход позволит обеспечить максимальную эффективность использования ВТ в учебном процессе.

Сейчас школы оснащаются разнообразной вычислительной техникой. В качестве графических устройств ввода-вывода могут использоваться алфавитно-цифровые дисплеи (с размерами экрана 16—25 строк по 40—80 символов) и алфавитно-цифровые печатающие устройства с числом символов в строке от 80 до 133. Эти устройства позволяют создать изображения, состоящие из единичных графических элементов — ЕГЭ. Под ЕГЭ будем понимать любой символ, который может быть выведен на экране дисплея или листе АЦПУ; положение его определяется двумя координатами — номером строки и позицией в строке.

Для создания ЕГЭ можно использовать операторы вывода любого языка программирования, например: WRITELN в Паскале, WRITE и FORMAT в Фортране, PRINT в Бейсике и т. д.

Задача 1. Построить ЕГЭ в заданном месте экрана (листа АЦПУ), используя только операторы вывода.

На начальном этапе решение этой задачи основано на организации пропуска нужного количества строк и позиций в строке; отсчет строк на экране дисплея или листе АЦПУ ведется от первой свободной, а нумерация позиций в строке — с крайней левой.

Решение задачи 1 может быть записано в виде:

Программа на Фортране

```
      PROGRAM POINT
C ПОСТРОЕНИЕ ЕГЭ —*—
C  С КООРДИНАТАМИ X=10, Y=10
      WRITE (K, 1)
1     FORMAT (X, 9(/), 9X, '*')
      STOP
      END
```

Программа на Бейсике

```
10 REM ПОСТРОЕНИЕ ЕГЭ —*—
20 REM С КООРДИНАТАМИ X=10, Y=10
30 PRINT\PRINT\PRINT\PRINT
   PRINT
40 PRINT\PRINT\PRINT\PRINT
50 PRINT TAB (9) '*'
60 END
```

Здесь и далее K (в операторе WRITE на Фортране) — номер логического устройства вывода.

На основе ЕГЭ можно строить различные по сложности изображения, в том числе движущиеся на экране дисплея или полутоновые на АЦПУ (за счет многократного прохождения печатающей головки по строке). Все это поддерживает интерес учащихся к новому предмету, стимулирует творческий поиск, дает им возможность использования структур языков программирования.

Рассмотрим конкретный подход к обучению программирования в курсе информатики на основе решения графических задач. Сначала будем использовать возможности только операторов вывода.

Создание простейших изображений — домика, ослика, кораблика и т. д. — как совокупность ЕГЭ проиллюстрировано в [2]. Подобные задачи демон-

стрируют возможности операторов вывода. Подробнее рассмотрим способы построения изображений простейших геометрических фигур: отрезка, окружности, дуги и т. п.

Задача 2. Построить изображение:
а) вертикального отрезка;
б) горизонтального отрезка.
Длину отрезка в задаче будем задавать в режиме диалога. Для решения этой задачи потребуется знание оператора цикла

Программа на Фортране

```
      PROGRAM STOLB1
C ПОСТРОЕНИЕ ВЕРТИКАЛЬНОГО
C ОТРЕЗКА ПРОИЗВОЛЬНОЙ ДЛИНЫ
C В ЗАДАННОМ МЕСТЕ
      WRITE (K, 5)
      READ (K, 10) L
C ЦИКЛ ПО ДЛИНЕ ОТРЕЗКА
      DO 20 I=1, L
   20 WRITE K, 30
      STOP
    5 FORMAT (X, 'ВВЕДИТЕ ДЛИНУ ОТ-
     РЕЗКА')
   10 FORMAT (I4)
   30 FORMAT (40X, '*')
      END
```

Программа на Бейсике

```
10  REM ПОСТРОЕНИЕ ВЕРТИКАЛЬ-
    НОГО
20  REM ОТРЕЗКА ПРОИЗВОЛЬНОЙ
30  REM ДЛИНЫ В ЗАДАННОМ
40  REM МЕСТЕ
50  PRINT 'ВВЕДИТЕ ДЛИНУ ОТРЕЗКА'
60  INPUT L
70  REM ЦИКЛ ПО ДЛИНЕ ОТРЕЗКА
80  FOR I=1 TO L
90  PRINT TAB(39) '*'
100 NEXT I
110 END
```

Эффективное построение изображения горизонтального отрезка произвольной длины требует использования специальных символов управления печатью. Приведем пример программы построения горизонтального отрезка.

Программа на Фортране

```
      PROGRAM LINE
C ГОРИЗОНТАЛЬНЫЙ ОТРЕЗОК
C ПРОИЗВОЛЬНОЙ ДЛИНЫ
      WRITE (K, 5)
      READ (K1, 10) L
C ЦИКЛ ПО ДЛИНЕ ОТРЕЗКА
      DO 20 I=1, L
   20 WRITE (K2, 30)
      STOP
    5 FORMAT (X, 'ВВЕДИТЕ ДЛИНУ ОТ-
     РЕЗКА')
   10 FORMAT (I4)
   30 FORMAT ('+', '*')
      END
```

Программа на Бейсике

```
10 REM ГОРИЗОНТАЛЬНЫЙ ОТРЕЗОК
15 REM ПРОИЗВОЛЬНОЙ ДЛИНЫ
20 PRINT 'ВВЕДИТЕ ДЛИНУ ОТРЕЗКА'
25 INPUT L
30 REM ЦИКЛ ПО ДЛИНЕ ОТРЕЗКА
35 FOR I=1 TO L
40 PRINT '*';
45 NEXT I
50 END
```

Отметим, что в операторе FORMAT для организации выдачи на различные устройства вывода используют разные управляющие символы. Приведенный вариант программы на Фортране соответствует выводу на АЦПУ. Для построения изображения горизонтального отрезка на экране дисплея оператор с номером 30 необходимо заменить на
30 FORMAT (⌧ , '*').

Задача 3. Построить изображение горизонтального отрезка длины L с координатами левого конца X и Y. Значения X, Y и L задавать с терминала в режиме диалога.

Задача 4. Нарисовать:
а) набор равноудаленных от ближайших соседей вертикальных отрезков;
б) прямоугольную рамку фиксированных размеров;
в) лестницу с параллельными сторонами;
г) забор.

Для решения этой задачи эффективно пользоваться подпрограммами.

Подпрограмма на Фортране

```
      SUBROUTINE GOD (I)
C ПРОГРАММА ПРОПУСКА
C ЗАДАННОГО ЧИСЛА СТРОК I⩾1
      DO 1 I1=1, I
    1 WRITE (K, 2)
    2 FORMAT (X)
      RETURN
      END
      SUBROUTINE GRED (I, J)
C С ПОДПРОГРАММА ПРОПУСКА
C В СТРОКЕ ЗАДАННОГО ЧИСЛА
C ПОЗИЦИЙ I⩾1 И ВЫВОДА
```

```
C ОТРЕЗКА ДЛИНЫ J⩾1
      DO 1 I1=1, I
    1 WRITE (K, 3)
      DO 2 J1=1, J
    2 WRITE (K, 4)
    3 FORMAT ('+', '␣')
    4 FORMAT ('+', '*')
      RETURN
      END
```

Подпрограмма на Бейсике

```
1000 REM ПОДПРОГРАММА ПРОПУСКА
1005 REM ЗАДАННОГО ЧИСЛА
1010 REM СТРОК I⩾1
1015 FOR I1=1 TO I
1020 PRINT
1025 NEXT I1
1030 RETURN
1000 REM ПОДПРОГРАММА ПРОПУСКА
1005 REM В СТРОКЕ ЗАДАННОГО ЧИС-
     ЛА
1010 REM ПОЗИЦИЙ I⩾1 И ВЫВОДА
1015 REM ОТРЕЗКА ДЛИНЫ J⩾1
1020 FOR I1=1 TO I
1025 PRINT '␣';
1030 NEXT I1
1035 FOR J1=1 TO J
1040 PRINT '*';
1045 NEXT J1
1050 RETURN
```

В качестве примера разберем задачу 4б. Ее решение основано на разбиении изображения на два горизонтальных отрезка и пару вертикальных. Отметим, что в программе перед построением изображения необходимо организовать проверку соответствия вводимых размеров рамки экрана (или листа АЦПУ). Для этого потребуется знание условного оператора.

```
      PROGRAM PAMKA
C ПРЯМОУГОЛЬНАЯ РАМКА С
C ЗАДАННЫМИ КООРДИНАТАМИ
C ВЕРХНЕГО ЛЕВОГО УГЛА
C И РАЗМЕРАМИ СТОРОН
      WRITE (K, 1)
      READ (K1, 2) IX, IY
      WRITE (K, 3)
      READ (K1, 2) IH, ID
C ПРОВЕРКА СООТВЕТСТВИЯ РАЗМЕ-
РОВ
C РАМКИ РАЗМЕРАМ ЭКРАНА
      IF((IH+IX).GT.24) IH=24—IX
      IF((ID+IY).GT.80) ID=80—IY
C ПРОРИСОВКА РАМКИ
      DO  5 I=1, IX—1
    5 WRITE (K, 100)
      CALL LINE (IY—1, ID)
      DO 6 I=1, IH
      CALL GOR (IY—1)
      WRITE (K, 102)
      DO 7 J=1, ID—2
    7 WRITE (K, 101)
    6 WRITE (K, 102)
      CALL LINE (IY—1, ID)
      STOP
    1 FORMAT (X, 'ВВЕДИТЕ КООР-
   * ДИНАТЫ ВЕРХНЕГО ЛЕВОГО
   * УГЛА')
    3 FORMAT (X, 'ВВЕДИТЕ ДЛИНЫ
   *  ВЕРТИКАЛЬНОЙ И ГОРИЗОН-
   *  ТАЛЬНОЙ СТОРОН')
    2 FORMAT (2I3)
  100 FORMAT (X, /)
  101 FORMAT (X, '_', ☒)
  102 FORMAT (X, '*', ☒)
      END
      SUBROUTINE GOR (N)
C УСТАНОВКА МАРКЕРА В N-Ю ПОЗИ-
ЦИЮ
C ФИКСИРОВАННОЙ СРОКИ
      DO 1 I=1, N
    1 WRITE (K, 2)
    2 FORMAT (X, '␣', ☒)
      RETURN
      END
      SUBROUTINE LINE (N, L)
C ПРОРИСОВКА ОТРЕЗКА ДЛИНЫ L
C N-Й
C ПОЗИЦИИ ФИКСИРОВАННОЙ СТРОКИ
      CALL GOR (N)
      DO 1 I=1, L
    1 WRITE (K, 2)
    2 FORMAT (X, '*', ☒)
      RETURN
      END
   10 REM РАМКА С ЗАДАННЫМИ
   20 REM КООРДИНАТАМИ
   30 REM ВЕРХНЕГО ЛЕВОГО УГЛА И
   40 REM РАЗМЕРАМИ СТОРОН
   50 PRINT 'ВВЕДИТЕ КООРДИНАТЫ
      X, Y'
   60 INPUT X, Y
   70 PRINT 'ВВЕДИТЕ РАЗМЕРЫ РАМ-
      КИ:'
   80 PRINT 'ВЫСОТУ H И ШИРИНУ D'
   90 INPUT H, D
  100 REM ПРОВЕРКА СООТВЕТСТВИЯ
  110 REM РАЗМЕРОВ РАМКИ И ЭКРАНА
  120 IF N+X>24 THEN H=24—X
  130 IF D+Y>=80 THEN D=79—Y
  140 REM ПРОРИСОВКА РАМКИ
  150 FOR I=1 TO X—1
  160 PRINT\NEXT I
  170 GOSUB 240
  180 PRINT
  190 FOR I=1 TO H
  200 PRINT TAB (Y); '*'; TAB(Y—1+D);
      '*'
  210 NEXT I
```

```
220 GOSUB 240
230 END
240 REM ПРОРИСОВКА ОТРЕЗКА
250 REM ДЛИНЫ D С Y-Й ПОЗИЦИИ
260 REM ФИКСИРОВАННОЙ
270 REM СТРОКИ
280 PRINT TAB(I—1+Y); '*'
290 NEXT I
300 RETURN
```

Рассмотрим другой подход к созданию изображения на экране — метод заочного рисования. Его можно применить после знакомства с понятием массива. Картинка (или ее элемент — одна строка экрана или листа АЦПУ) создается в памяти ЭВМ путем заполнения двумерного массива (например, размерности 24×80), номер строки и номер столбца которого совпадают с координатами ЕГЭ на экране — номером строки и позицией в строке. Полученный образ картинки выводится на экран или лист АЦПУ простой распечаткой массива. Проиллюстрируем это задачей.

Задача 5. Используя метод заочного рисования, построить отрезок с задаваемыми в режиме диалога координатами концов. (Рекомендуется использовать уравнение прямой: $\frac{Y-Y1}{Y2-Y1} = \frac{X-X1}{X2-X1}$.) Заметим, что без использования массивов решить эту задачу достаточно сложно.

Подпрограмма на Фортране

```
      SUBROUTINE LINE(IX1, IY1,
     *IX2, IY2, ISYM, SCRN)
C ПОДПРОГРАММА ПОСТРОЕНИЯ ПРЯ-
МОЙ ЛИНИИ
C МЕТОДОМ ЗАОЧНОГО РИСОВАНИЯ
C ВХОДНЫЕ ДАННЫЕ: IX1, IY1, IX2,
IY2 —
C КООРДИНАТЫ ТОЧЕК ОТРЕЗКА
C ISYM — СИМВОЛ ЕГЭ
C SCRN — РАБОЧИЙ МАССИВ
      BYTE SCRN(24, 80), ISYM
      ISTEP=1
      IF(IX1.EQ.IX2) GOTO 10
      IF(IX1.GT.IX1) ISTEP=—1
C ЦИКЛ ВЫЧИСЛЕНИЯ КООРДИНАТ
C ТОЧЕК, ПРИНАДЛЕЖАЩИХ ОТРЕЗКУ
      DO 5 IX=IX1, IX2, ISTEP
      IY=(IX—IX1)*(IY2—IY1)/
     *(IX2—IX1)+IY1
    5 SCRN(IY, IX)=ISYM
      RETURN
C ПОСТРОЕНИЕ ВЕРТИКАЛЬНОЙ ЛИ-
НИИ
   10 IF(IY1.GT.IY2) ISTEP=—1
      DO 20 IY=IY1, IY2, ISTEP
   20 SCRN(IY, IX1)=ISYM
      RETURN
      END
```

Подпрограмма на Бейсике

```
 5 REM РИСУЕМ ЛИНИЮ
 6 DIM D%(12, 80)
 7 PRINT 'ВВЕДИТЕ КООРДИНАТЫ'
 8 PRINT 'ПЕРВОЙ ТОЧКИ — X1%, Y1%'
 9 INPUT X1%, Y1%
10 PRINT 'ВТОРОЙ ТОЧКИ — X2%, Y2%'
11 INPUT X2%, Y2%
12 REM ОЧИСТКА ЭКРАНА
13 FOR I=1 TO 12
14 FOR J=1 TO 80
15 D%(I, J)=' '
16 NEXT J\NEXT I
20 S%=1
21 D%(Y1%,  X1%)='*'\Y3%=Y1%
23 IF X1%=X2% GOTO 100
24 FOR X%=X1% TO X2% STEP S%
26 Y%=(X%—X1%)*(Y2%—Y1%)/
   (X2%—X1%)+Y1%
28 IF Y3%=Y% GOTO 55
30 D%(Y%, X%)='*'
32 Y3%=Y%
55 NEXT X%
60 FOR I=1 TO 12
61 FOR J=1 TO 80
65 PRINT D%(I, J)
67 NEXT J\PRINT\NEXT I
70 END
100 REM ВЕРТИКАЛЬ
105 IF Y1%>Y2% THEN S%=—1
110 FOR Y%=Y1% TO Y2% STEP S%
120 D%(Y%, X%)='*'
125 NEXT Y%
130 FOR I=1 TO 12\FOR J=1 TO 80
135 PRINT D%(I, J) \NEXT J
140 PRINT\NEXT I
145 END
```

Задача 6. Решить задачи 3 и 4 с использованием подпрограммы LINE.

Задача 7. Составить программу, рисующую на экране дисплея горизонтальную змейку с числом звеньев, задаваемым в режиме диалога.

Задача 8. Составить программу, печатающую ЕГЭ в шахматном порядке (в строках с четными номерами — в четных позициях, в строках с нечетными номерами — в нечетных).

Динамические изображения

Рассматриваемые до сих пор методы и приемы предполагали последовательное заполнение экрана построчно сверху вниз. Такой подход является приемлемым только для создания статических изображений. Построение динамических картинок — движущихся в любом направлении объектов — требует использования встроенных возможностей дисплея по перемещению маркера. Напомним, что маркер — это светящаяся область на экране; перемещая его, можно управлять позицией вывода знаков на экран. Для этого применяются специальные подпрограммы, устанавливающие маркер в позицию экрана с заданными координатами. При работе с Фортраном используется подпрограмма MARKER, описание которой для машин серии СМ ЭВМ (ДВК, СМ-3, СМ-4, СМ1420) в операционной системе ОС РВ приведено в приложении.

Обращение к подпрограмме имеет вид:

CALL MARKER (K1, K2),

где K1, K2 — числовые константы или имена переменных, определяющие номер строки и номер позиции в строке соответственно. Дополнительной возможностью подпрограммы является очищение экрана с помощью вызова CALL MARKER ('DS').

Задача 9. Составить программу перемещения ЕГЭ по экрану слева направо.

Решение задачи на Фортране будет выглядеть так:

```
      PROGRAM DYNAMIC
С ПРОГРАММА ПЕРЕМЕЩЕНИЯ ЕГЭ
ИЗ ТОЧКИ
С ЭКРАНА С КООРДИНАТАМИ (10, 10)
ПО
С ОСИ X ДО ПРАВОЙ ГРАНИЦЫ ЭКРАНА
      CALL MARKER ('DS')
      JY=10
      DO 1 JX=10, 80
      CALL MARKER (JY, JX)
    1 WRITE(K, 5)
      CALL MARKER(JY, JX)
      WRITE(K, 10)
      STOP
    5 FORMAT(X, '*')
   10 FORMAT(2X)
      END
```

Графический пакет

Обладая возможностями по перемещению маркера, можно перейти к созданию простейшего набора графических подпрограмм. Такой набор позволит в дальнейшем существенно упростить создание изображений, используя принцип модульности языков программирования.

Задача 10. Используя подпрограмму MARKER, написать следующие подпрограммы, которые составят возможную основу графического пакета:

а) построение точки по заданным координатам;

б) построение линии (по аналогии с задачей 6);

в) построение окружности; входные параметры: координаты центра, величина радиуса, символ ЕГЭ;

г) построение дуги; входные параметры: координаты начала, центра и конца, ЕГЭ (возможно и другое задание входных параметров).

Этот пакет можно использовать при решении следующих учебных задач.

Задача 11. Построить на экране изображение: квадрата, треугольника, Чебурашки, эмблемы Олимпийских игр и т. д.

Задача 12. Составить программу перемещения по экрану окружности в произвольном направлении.

Задача 13. Составить программу, моделирующую на экране:

а) движение обруча по наклонной плоскости;

б) качающийся маятник;

в) деление клетки;

г) образование углеводов из жиров.

Задача 14. Написать диалоговую программу печати объявлений на АЦПУ различными шрифтами: мальнькие, большие, прописные, строчные, русские, английские буквы.

Задача 15. Печать таблиц, графиков, сводок и другой административной документации для школы.

Задача 16. «Попади в корзину!» Возможный сценарий: на экране в фиксированном месте нарисована «баскетбольная корзина». Через определенный интервал времени на экране появляется изображение мяча — точка. Цель

игры: управляя изменением угла удара и его силы, забросить «мяч» в «корзину». Во время игры можно вести подсчет очков.

Задача 17. «Луноход».

Цель игры: перемещая спасательный луноход, за минимальное время попасть в заданную точку — место аварии космического корабля. Направление и скорость лунохода определяются в результате правильного выполнения элементарных упражнений по грамматике, математике и другим предметам. Ошибочное решение затрудняет продвижение.

Методика, изложенная в статье, и задачи могут использоваться как при изучении информатики, так и при создании программной поддержки учебных курсов других предметов. Фрагменты таких программ должны использоваться в качестве индивидуальных практических заданий учащимся для самостоятельной разработки.

Литература

1. *Форсайт Р.* Паскаль для всех. М.: Машиностроение, 1986.
2. *Кетков Ю. Л.* Программирование на Бейсике. М.: Статистика, 1978.
3. *Салтыков А. И., Семашко Г. Л.* Программирование для всех М.: Наука, 1986. С. 175.

ПРИЛОЖЕНИЕ.
Подпрограммы управления позицией вывода.

1. *Подпрограмма для работы с Фортраном*

```
          .TITLE MARKER
          .MCALL QIOW$S
; ПОДПРОГРАММА НА МАКРОАССЕМБЛЕРЕ
; ВЫЗОВ ИЗ ФОРТРАНА: CALL MARKER (НОМЕР
; СТРОКИ, НОМЕР ПОЗИЦИИ В СТРОКЕ)
;
;
MARKER::
          MOVB    #'Y,ESCP+1
          MOV     #4,R3          ; ДЛИНА УПРАВЛЯЮЩЕЙ
                                   ПОСЛЕДОВАТЕЛЬНОСТИ
          CMP     #1,(R5)+
          BNE     1$             ; ЗАПРОС НА ПЕРЕМЕЩЕНИЕ МАРКЕРА
          MOV     #2,R3
          MOVB    #'J,ESCP+1     ; ЗАПРОС НА УДАЛЕНИЕ СТРОКИ ИЛИ
                                   ОЧИСТКУ
          CMP     #'DL,@(R5)     ; ЭКРАНА
          BNE     OUT            ; ОЧИСТКА ЭКРАНА
          MOVB    #'K,ESCP+1     ; УДАЛЕНИЕ СТРОКИ
          BR      OUT
1$:       MOVB    @(R5),LINE     ; НОМЕР СТРОКИ
          BMI     ERROR          ; ОШИБКА. НОМЕР СТРОКИ
                                   ОТРИЦАТЕЛЕН
          ADD     #40,LINE
          TST     (R5)+
          MOVB    @(R5),R4       ; ПОЗИЦИЯ В СТРОКЕ
          ADD     #40,R4
          MOVB    R4,COLUMN
;
OUT:      QIOW$S  #IO.WLB,#5,#1,,#IOSB,,<#ESCP,R3>
                                 ; ОБРАЩЕНИЕ К ДРАЙВЕРУ
                                   ВВОДА/ВЫВОДА
          TSTB    IOSB           ; БЫЛА ЛИ ОШИБКА ПРИ ОБРАЩЕНИИ
                                   К ТЕРМИНАЛУ?
          BGE     EX
;
ERROR:    0                      ; ОСТАНОВ ПО ОШИБКЕ
```

```
EX:     RETURN                  НОРМАЛЬНЫЙ ВОЗВРАТ
;
ESCP:   .BYTE   33,'Y           ; УПРАВЛЯЮЩАЯ ПОСЛЕДОВАТЕЛЬНОСТЬ
                                  ПОЗИЦИОНИРОВАНИЯ
LINE:   .BYTE   0               ; НОМЕР СТРОКИ
COLUMN: .BYTE   0               ; ПОЗИЦИЯ В СТРОКЕ
IOSB:   0,0
;
        .END
```

Для использования подпрограммы необходимо проделать следующие действия:
1) с помощью редактора текстов создать на диске файл с именем MARKER.MAC;
2) оттранслировать полученный файл с помощью команды:

MAC MARKER=MARKER;

3) при формировании готовой пользовательской задачи всегда следует включать в загрузочный модуль файл, полученный в п.2 (например в ОС РВ: ТВК имя задачи пользователя=MARKER, список файлов пользователя).

2. Подпрограмма для работы с Бейсиком

```
1000 REM УСТАНОВКА МАРКЕРА В ТОЧКУ (X%, Y%)
1010 PRINT CHR$(27); 'H';
1015 IF D$<>'YES' GOTO 1025
1020 PRINT CHR$(27); 'J'
1025 IF X%=0 GOTO 1065
1030 FOR I=1 TO X%
1035 PRINT
1040 NEXT I
1045 IF Y%=0 GOTO 1065
1050 PRINT TAB(Y%);
1065 END
```


Примечание: при использовании Бейсика количество символов в строке (длина строки вывода) на экране дисплея регулируется оператором

A=TTYSET(255%,K),

где K — целая константа, такая, что K-1 есть требуемая длина строки.

КИМ В., СИНИЦЫН Е., КОНЕВ А., ЕМЕЛЬЯНОВ Ю.

«Барьер» — эффективное средство обучения

В своей речи на общесоюзном совещании преподавателей кафедр общественных наук М. С. Горбачев определил ряд основных принципиальных концепций обучения: эмоциональная интеллектуальность, диалог, диалектика, необходимость «радостных находок... открытий, по крайней мере для себя».

В рамках этих выдвинутых концепций в Новосибирске развивается новый метод обучения путем микрооткрытий. На его основе нами создана автоматизированная обучающая система «Барьер».

Известно, что «проблема программирования является сейчас узким местом на пути к всенародному применению микроэлектронной вычислительной техники» [1]. Достижение массовости программирования тесно связано с эффективными методами обучения. Существуют два варианта решения этой проблемы.

Первый — резкое увеличение числа квалифицированных преподавателей, что трудно достичь за короткое время при традиционных методах обучения. Существуют две основные категории специалистов, *потенциально* способных решить эту задачу: с одной стороны — это учителя средних школ, преподаватели СПТУ и техникумов и, с другой стороны — профессиональные про-

граммисты НИИ и ВЦ. Нельзя с уверенностью сказать, что преподаватель математики или физики, прошедший курсы повышения квалификации (максимум два месяца да еще и традиционным методом — лекции и практические занятия), способен набрать профессиональный опыт, достаточный для свободного ориентирования в специфике программирования. Остается еще армия программистов НИИ и ВЦ, но, увы, не каждый учитель способен быть программистом и, наоборот, программист — учителем.

Второй путь, достаточно эффективный во времени,— использование для обучения самих средств вычислительной техники. Иначе говоря — создание высокоэффективных автоматизированных обучающих систем (АОС), способных за кратчайшее время обучить началам программирования в объеме, определяемом формулой компьютерной грамотности [1]. Очень важно отметить высокую эффективность систем «учитель—АОС—учащиеся» и «программист—АОС—учащиеся». В первом случае обучающая система компенсирует недостаточный опыт работы преподавателя с ЭВМ, во втором — недостаточный педагогический опыт программиста или инженера.

«Барьер» сокращает сроки обучения, повышает успеваемость учащихся [2, 3]. Новая тема предъявляется обучаемым в виде сильно связанных смысловым содержанием микропроблем, сформулированных в виде наводящих вопросов. Истинный ответ на поставленный вопрос выдается обучающей системой только после ответа обучаемого, т. е. в момент максимальной концентрации его внимания, что обеспечивает прочное усвоение полученного фрагмента знаний. Эксперименты показали, что максимальная эффективность обучения достигается при такой трудности вопросов, когда приблизительно 75 % правильных ответов обучаемый находит самостоятельно. Такая дозировка трудности вопроса обеспечивает эмоциональную мотивацию обучаемых, стимулируя их продолжать учебу [3].

Решение поставленных вопросов, ответы на которые обусловлены догадками или микрооткрытиями обучаемых, облегчается ассоциациями и аналогиями.

Вторым принципиальным моментом технологии обучения является использование опорных схем В. Ф. Шаталова, формируемых обучающей программой.

Основные принципы разработанной АОС «Барьер»:

совмещение функций обучения и контроля;

стимуляция творческого мышления за счет догадок или микрооткрытий;

адаптируемость системы обучения с помощью вариации трудности вопросов;

наличие быстрой обратной связи при обучении в режиме диалога;

многократное повторение определенных связей и понятий, отражающих суть проблемы изучаемой темы;

приведение учащегося к разрешению противоречия.

АОС «Барьер» внедрена в терминальном классе Новосибирского техникума электронных приборов на базе мини-ЭВМ «Электроника-100—25» и обеспечивает одновременное обучение 24 человек. Программное обеспечение АОС отличается достаточной гибкостью и универсальностью. Для иллюстрации ее возможностей, кроме курса программирования, разработан ряд уроков по физике, химии, политэкономии, электро- и радиоматериалам, теории вероятностей, теории алгоритмов и формальных языков, информатике, высшей математике.

Обучение на АОС «Барьер» прошли инженерно-технические работники, учащиеся техникума и школ, студенты вузов, учителя школ. В настоящее время проведено более 1500 занятий.

Система применялась при обучении программированию на языке Фортран учителей школ (состав группы: директора, завучи, учителя математики, физики, химии, биологии, рисования, начальных классов). Оказалось, что в течение 24 час обучаемые получили навыки самостоятельного составления программ, ввода их в ЭВМ, отладки и получении результата в среде операционной системы реального времени. Следует заметить, что слушатели до начала обучения не имели ни малейшего представления ни об ЭВМ, ни о программировании.

Класс разбивается на несколько подгрупп по 2—4 человека, подбор определяется на основе личной симпатии, при наличии лидера, оппонента, с учетом темперамента.

Каждая группа осваивает сценарий с помощью «мозгового штурма» независимо от другой, поэтому темп у них разный.

После того как один из членов группы прочитал вопрос, начинается «мозговой штурм». Когда коллективная догадка найдена, ведущий открывает ответ. И если ответ сценария совпал с догадкой группы, то против соответствующего номера ответа ставится «плюс». Если отгадка неверна — «минус».

Учитель анализирует количество «плюсов» и «минусов». Это необходимо для того, чтобы определить оптимальность сценария с эмоциональной точки зрения. Если встречается три «минуса» подряд, то изменяется трудность вопросов путем введения дополнительных наводящих вопросов между ними. Оптимальным по критерию эмоциональной стимуляции считается сценарий, если из 35 вопросов на 4—5 не удается найти правильный ответ.

Как правило, трудно угадать основные понятия. В этом случае учащийся усваивает эти понятия, прочитывая ответ сценария в момент максимальной интеллектуальной напряженности.

После 4—6 безмашинных занятий, когда учащиеся освоили основные операторы языка, рекомендуется посещение терминального класса для отладки программ на ЭВМ. Такая циклическая процедура обучения, как показал опыт, способствует успешному овладению навыками программирования.

Интересно заметить, что проведенные таким образом занятия в пионерских лагерях с последующим эпизодическим выходом в терминальные классы, где практически в это время отсутствовала загрузка АОС «Барьер», дали мощный импульс последующей кружковой работе по информатике. В настоящее время в Новосибирске есть три кружка (по 40 учащихся), работающих по данной методике.

Для иллюстрации безмашинного метода обучения прилагается сценарий урока по Бейсику (первое занятие). Весь первоначальный цикл обучения Бейсику содержит шесть таких занятий.

Отлаживать АОС «Барьер» помогали учащиеся I курса техникума (участники кружка). За три месяца они смогли выйти на уровень самостоятельной работы (программы на языке Фортран — 200—250 строк и использование подпрограмм на ассемблере). В настоящее время программированию на языках высокого уровня с помощью АОС в техникуме учатся свыше 250 человек.

АОС «Барьер» базируется на операционной системе мини-ЭВМ «Электроника-100—25» (RSX—IBM). Однако авторами предпринимаются определенные усилия для применения АОС «Барьер» на других мини- и микро-ЭВМ (в частности, идет разработка варианта для микро-ЭВМ «Электроника-60»).

Вполне понятно, что из-за отсутствия в большинстве школ, СПТУ вычислительной техники эффективно применять систему в массовом масштабе пока трудно. Однако метод микрооткрытий позволяет эмоционально стимулировать [3] учебу в рамках эпизодических выходов на ЭВМ.

В этих случаях мы применяем диалог учителя со всем классом и обучение с помощью управляемого «мозгового штурма», когда используется напечатанный на бумаге или карточках диалог. Расположение на бумаге вопросов и ответов основывается на следующей схеме. Каждая пара «вопрос-ответ» отделена от предыдущей и последующей двумя чертами. Вопрос от ответа отделен одной чертой.

Литература

1. Микропроцессорные средства и системы. М., 1986. № 1. С. 91—92.

2. *Ким В. В., Синицын Е. С., Конев А. Д.* Автоматизированная обучающая система, стимулирующая творческое мышление студентов // XXIX областная научно-техническая конференция: Тезисы докладов. Новосибирск, 1986. С. 35.

3. *Синицын Е. С., Конев А. Д., Ким В. В.* Об эмоциональной стимуляции в методе микрооткрытий // Научная организация учебного процесса. Новосибирск, 1986.

ЯЗЫК ПРОГРАММИРОВАНИЯ БЕЙСИК
(1-е занятие)

1. Что такое язык, например русский или английский, для чего он служит?

Средство общения между людьми

2. Для чего служит язык программирования на ЭВМ

Для общения между человеком и ЭВМ

3. Что такое инструкция по пользованию бытовым прибором, например телевизором?

Порядок, последовательность действий:

1 действие
2 действие
3 действие
. . .

4. Что дает командир во время боя своим бойцам?

Команды, приказы

5. Из чего состоит программа для ЭВМ?

Из последовательности команд:

1 команда
2 команда
3 команда
. . .

6. Что такое команда?

Приказ выполнить действие

7. Как команды расположены в программе? Хаотично?

Нет, по порядку, упорядоченно

8. В каком блоке ЭВМ хранится информация?

В блоке памяти

9. Как вы думаете, сколько ячеек памяти занимает 1 число?

1 ячейку: | 5 |

10. Видим очередь в телефонную будку. Когда один закончил звонить, что он делает?

Выходит из будки

11. Кто затем оказывается в будке?

Следующий

12. Когда команда посылает число 5 в ячейку памяти, а содержимое ячейки было равно ∅, какое число там будет?

Число 5; 5⟶ | ∅ | | 5 |

13. Если после этого снова в ту же ячейку послать число 8, то какое в ней будет число?

8

14. Что произойдет с числом 5 в этой ячейке?

Оно будет стерто, уничтожено:

8⟶ | 5 | | 8 |

15. Какие числа напечатают программы «ВОВА» и «ЛЕНА», вычисляя Z?

```
1 NEW BOBA          1 NEW ЛЕНА
2 Z=2+3             2 Z=1+8—2
3 PRINT Z           3 PRINT Z
4 STOP              4 STOP
```

5 7

16.

```
1 NEW BOBA
2 Z=1+5
3 PRINT Z
4 STOP
```

Как вы полагаете, какая из строк (команд) этой программы отдает приказ закончить работу программы?

4-я команда

17.

```
1    BOBA
2  =1+5
3
4
```

Какая команда производит вычисление?

2-я команда: Z=1+5

18.

```
1 NEW BOBA
2 Z=1+5
3 PRINT
4 STOP
```

Как вы думаете, какая строка (команда) этой программы отдает приказ печатать Z?

3 PRINT Z

19.

```
1 NEW BOBA          1 NEW ЛЕНА
2 Z=1+5             2 Z=1+8—2
3 PRINT Z           3 X=2+4
4 STOP              4 PRINT Z
                    5 PRINT XX
                    6 STOP
```

Как вы считаете, сколько всего команд в программе «ВОВА» и сколько в программе «ЛЕНА»?

4 команды 6 команд

20. Как вы думаете, что делает 1-я команда этих программ?

Дает программе название (имя)

21. Зачем программе имя?

Чтобы отличить ее от других программ

22. Что делает 2-я команда программы?

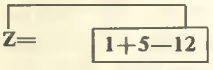
```
1 NEW BOBA
2 Z=1+5
3 PRINT Z
4 STOP
```

Вычисляет и присваивает Z значние 6

23. В чем отличие присваивания от вычисления?

В ЭВМ сначала вычисляется арифметическое выражение, стоящее справа от знака =

А затем вычисленное значение посылается в ячейку памяти Z



24. Что делает последняя команда этой программы?

```
1 NEW BOBA
2 Z=1+5
3 PRINT Z
4 STOP
```

Останавливает выполнение программы

25. Какие числа напечатает программа:

```
  NEW SUMMA
2 Z=5+2
3 X=1+3+5
4 PRINT Z, X
5 STOP
```

7, 9

26. Есть две программы:

1-я программа:	2-я программа:
NEW BOBA	NEW ЛЕНА
2 X=∅	2 X=1
3 X=1	3 X=∅
4 PRINT X	4 PRINT X,X
5 STOP	5 STOP

Какое число напечатает 1-я программа?

1

27. Какие два числа напечатает 2-я программа?

1-я программа:	2-я программа:
NEW BOBA	NEW ЛЕНА
2 X=0	2 X=1
3 X=1	3 X=∅
4 PRINT X	4 PRINT X,X
5 STOP	5 STOP

∅, ∅

28. Перед вами программа и ячейка памяти X:

```
  PROGRAM SUM
2 X=1
3 X=∅
4 PRINT X,X
5 STOP
```

X [ ]

Какое число запишется в ячейку X, когда выполнится 2-я команда этой программы?

Запишется 1: X [1]

29.

```
  NEW ЛЕНА
2 X=1
3 X=0
4 PRINT X,X
5 STOP
```

Какое число запишется в ячейку X, когда выполнится 3-я команда этой программы?

Запишется ∅: X [∅]

30. Что произошло с предыдущим значением X, равным 1?

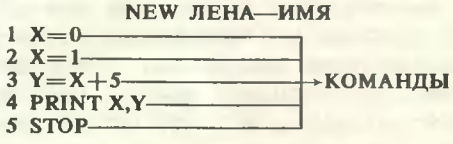
Оно стерто, уничтожено последующим значением, равным ∅

31.

```
  NEW BOBA
2 X=∅
3 X=1
4 PRINT X,X,X
5 STOP
```

Какие три числа напечатает эта программа?

1, 1, 1



32. Какие числа напечатает эта программа?

1, 6

33. Составьте программу с именем ЛЕНА, которая вычисляет Z=X+1725 при X=6237 и печатает значение Z

Программу проверит преподаватель

Конец занятия

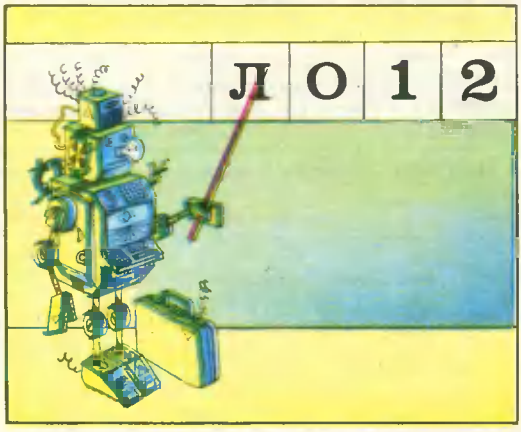
ДУВАНОВ А.
Благовещенский педагогический институт

Информатика без ЭВМ

Казалось бы, не стоит тратить усилий на подогрев интереса школьников к информатике. Эта дисциплина предполагает изучение работы и саму работу на такой таинственной технике, как электронно-вычислительная машина. Кто же не знаком с книгами по робототехнике А. Азимова, фантастическими возможностями и перспективами, которые дают человеку компьютер, робот! Однако в большей части школ у учеников начинает появляться чувство, что их здорово надувают. Оно обусловлено отсутствием ВТ в школах, недоверием к предмету у самих учителей и, самое главное, отсутствием достаточного количества примеров использования ВТ в ближайшем окружении.

По нашему мнению, повысить интерес школьника к предмету можно не столько описанием того, что может компьютер в принципе и что он уже где-то делает, сколько тем, что может получить школьник от ЭВМ или хотя бы воображаемого исполнителя прямо сейчас.

Заметим, что отсутствие у школьника доступа к ЭВМ делает изучение Бейсика дважды вредным. Во-первых, этот язык сам по себе непригоден для школы из-за своей неадекватности современной методологии программирования. Во-вторых, без машины Бейсик вообще теряет смысл, ибо пропадает главный его козырь — возможность исполнения программ на ЭВМ. Однако в тысячах школ десятиклассники изучают Бейсик и Рапиру, не имея доступа к машине. Нам кажется, что в таких школах надо и в X классе продолжать изучать методологию построения алгоритмов на базе алгоритмического языка, причем больше внимания уделять примерам из конкретного окружения ученика. Сам язык неплохо бы дополнить новыми структурами данных для удобства обработки информации, отличной от числовой. Традиционная направленность практики построения алгоритмов на числовые задачи, перенесенная в среднюю школу из высшей, вряд ли оправдана: малы и удельный вес вычислительных задач, решаемых сегодня на ЭВМ, и процент школьников, интересы которых связаны только с математикой.



Перенос задач из области математики в другую часто можно сделать простым изменением формулировки: вместо нахождения суммы элементов массива находить средний балл класса, вместо поиска максимума среди отвлеченных чисел заниматься поиском наивысшей температуры в таблице и т. д., конечно, не забыв дать и алгоритм, «очищенный» от приложений. Здесь все аналогично математическим абстракциям: неправильно учить, что производная — скорость, а интеграл — площадь, но важно, чтобы приложения были, до или после обобщений, но были. Иначе может пропасть интерес к предмету и сложиться впечатление, что цель программирования — отыскание максимума из трех чисел и подсчет количества нулей в массиве.

Повысить заинтересованность учащихся можно также использованием различных исполнителей, привлекающих своей средой обитания и достаточно широкими возможностями для творчества.

На основе опыта работы с новосибирскими школьниками автором совместно

с О. Г. Какаулиным разработана программная система «Джойстик», предназначенная для управления различными исполнителями на базе простого языка, включающего в себя три средства: вызов процедуры, развилку и цикл. Под процедурой понимается либо команда исполнителя (стандартная процедура), либо процедура пользователя. Сейчас система реализована на микроЭВМ «Искра-226», в состав ее входят два исполнителя: Таракан и Корректор.

«Джойстик» — операционная среда и одновременно язык программирования, рассчитанный на управление различными исполнителями. Программа записывается в диалоговом режиме рисованием на экране дисплея изображений, что повышает наглядность алгоритма и делает программы более понятными. Редактор системы позволяет проводить любые допустимые изменения в процедурах пользователя, просматривать листинги на экране или выводить их на графопостроитель. Библиотекарь системы позволяет записывать процедуры пользователя на диск, уравнивая их тем самым со стандартными. Система работает с пользователем в режиме модели многооконного интерфейса. Выбор необходимый команды производится с помощью оконных меню. «Джойстик» допускает только один способ программирования —структурный, поэтому специально учить структурированию программ не надо.

Основные синтаксические правила «Джойстика» на языке Бэкуса описываются так:

⟨программа⟩::=ПРОЦ[ЕДУРА] ⟨имя⟩
    [⟨тело программы⟩]
    КОН[ЕЦ]
⟨тело программы⟩::=⟨структура⟩ |
    ⟨тело программы⟩ ⟨структура⟩
⟨структура⟩::=⟨вызов процедуры⟩ |
    ⟨развилка⟩ | ⟨цикл⟩
⟨вызов процедуры⟩::=⟨имя⟩
⟨развилка⟩::=ЕСЛИ ⟨условие⟩
    [ТО ⟨вызов процедуры⟩]
    [ИНАЧЕ ⟨вызов процедуры⟩]
⟨цикл⟩::=ЦИКЛ ⟨параметр⟩
    ⟨вызов процедуры⟩

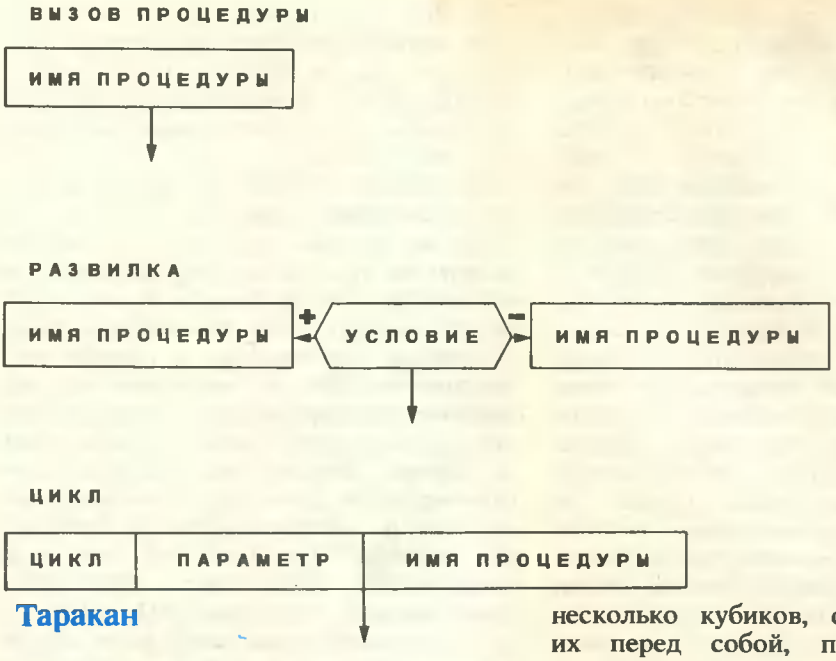
Именем может служить любой набор символов, начинающийся не с цифры. Параметр — целое число — задает количество повторений выполнения процедуры в цикле. Набор имен стандартных процедур, а также смысл условия в развилке задаются отдельно для каждого исполнителя.

Изображение структур на экране дисплея показано на рис. 1.

Среда каждого исполнителя состоит из двух листов. На первом отображается рабочее поле исполнителя и находится меню, позволяющее давать начальные установки исполнителю, запускать процедуры на счет и наблюдать за их выполнением, переходить ко второму листу, распечатывать каталог библиотеки на диске. Второй лист служит для формирования процедур и их редактирования, распечатки справок о процедурах, находящихся в данный момент в оперативной памяти. Лист имеет окна управляющих структур, СПИ (системы предписаний исполнителя), редактора и построения листинга. Работа в окнах, как и всюду в системе, состоит в перемещении световой метки клавишами управления курсора для выбора нужного раздела.

По исполнителям Таракан и Корректор написаны пособия, ориентированные непосредственно на школьников и предназначенные в первую очередь для организации безмашинного варианта преподавания. Экран дисплея заменяют модели сред исполнителей, а процессор — сами ребята, которые управляют исполнителями по написанным программам вручную. Пособие «Введение в программирование. Таракан и Тараканчик» (А. А. Дуванов, В. В. Немилостива, О. Г. Какаулин) в течение двух лет апробировалось в 20 сельских школах и городском кружке «Мой друг компьютер». Получены положительные отзывы учителей, ведущих факультативные занятия, и дополнительные заявки на это пособие. Сейчас по решению ученого совета Благовещенского пединститута оно издается для школ Амурской области тиражом 1,5 тыс. экземпляров.

Приведем краткое описание исполнителей и рассмотрим в качестве иллюстрации несколько задач.



Таракан

Идея этого исполнителя заимствована из книги Г. А. Звенигородского «Первые уроки программирования» (М.: Наука, 1985).

Среда обитания. Таракан может ползать по клетчатому полю и передвигать по нему кубики с разными надписями или рисунками на них. За один ход он сдвигается на одну клетку вверх, вниз, влево, вправо (но не по диагонали). Наткнувшись на один или несколько кубиков, он будет толкать их перед собой, пока не изменит направления движения. Считается, что поле обнесено по краям глубоким рвом. Таракан может спихнуть в ров кубики, но сам лезть туда боится.

Стандартные процедуры. ВВЕРХ ВНИЗ ВЛЕВО ВПРАВО

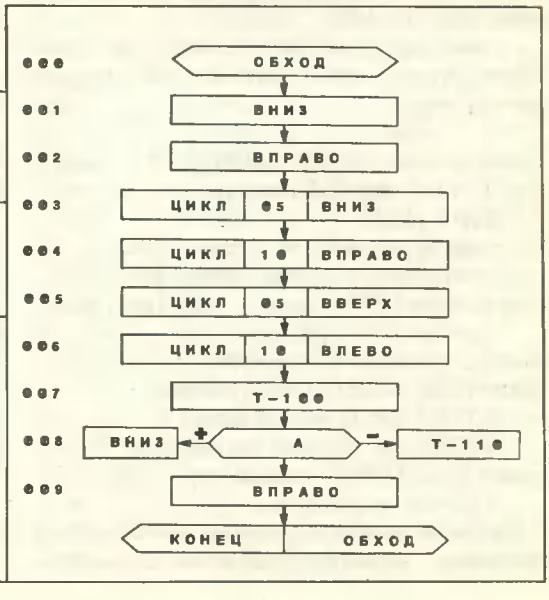
Структуры. Развилка. Считается, что некоторые кубики перевернуты надписями вниз. Таракан, толкая такой кубик, переворачивает его и видит надпись; развилка обеспечивает принятие реше-

```
ИСПОЛНИТЕЛЬ ТАРАКАН
0 ЧАС 37 МИН 15 СЕК
ФОРМИРОВАНИЕ ПРОЦЕДУР И РЕДАКЦИЯ

УПРАВЛЯЮЩИЕ СТРУКТУРЫ
  НАЧАЛО ПРОЦЕДУРЫ    ПРОЦЕДУРА
  ВЫХОД               РАЗВИЛКА
  РЕДАКЦИЯ            ЦИКЛ

СИСТЕМА ПРЕДПИСАНИЙ ИСПОЛНИТЕЛЯ
  ИМЯ ПРОЦЕДУРЫ       ВНИЗ
  ПУСТО               ВЛЕВО
  ВВЕРХ               ВПРАВО

РЕДАКТОР ПРОЦЕДУР
  ЛИСТИНГ             УДАЛЕНИЕ
  НОМЕР               ВСТАВКА
  КОНЕЦ               ЗАМЕНА
  ИСПРАВЛЕНИЕ         ГРАФ
```



ния в зависимости от увиденного. Условием развилки служит символ, который сличается с символом на перевернутом кубике. Если они идентичны, условие считается выполненным.

Цикл. Любая процедура, стандартная или пользователя, библиотечная или из памяти системы, может быть повторена в цикле от 0 до 99 раз.

Примеры задач для Таракана

1. Помогите Таракану построить дом (буква «о» на одном из перевернутых кубиков) (рис. 2).

Р е ш е н и е.

```
ПРОЦЕДУРА дом
    ВВЕРХ
      ЕСЛИ о
      ИНАЧЕ а
КОНЕЦ
ПРОЦЕДУРА а
    ВЛЕВО
    ВЛЕВО
    ЦИКЛ 3 ВВЕРХ
    ВПРАВО
    ВПРАВО
    ВНИЗ
КОНЕЦ
```

2. Поднимите Таракана по лесенке (рис. 3).

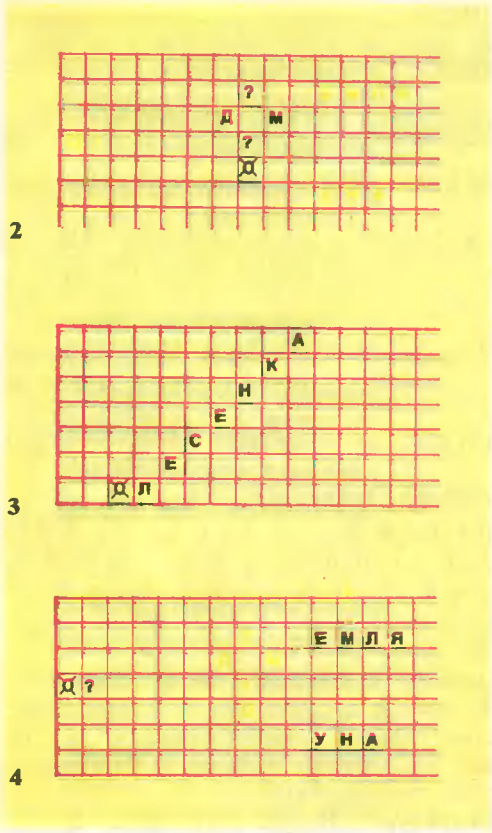
Р е ш е н и е.

```
ПРОЦЕДУРА лесенка
    ЦИКЛ 6 ступень
КОНЕЦ
ПРОЦЕДУРА ступень
    ВВЕРХ
    ВПРАВО
КОНЕЦ
```

3. Где находится Таракан: на Земле или на Луне? (На перевернутом кубике буква либо «з», либо «л».) (Рис. 4.)

Р е ш е н и е.

```
ПРОЦЕДУРА полет
    ВПРАВО
    ЕСЛИ з ТО а1
      ИНАЧЕ а2
КОНЕЦ
ПРОЦЕДУРА а1
    в
    ВНИЗ
    ВПРАВО
    ВВЕРХ
    ВВЕРХ
КОНЕЦ
```




```
ПРОЦЕДУРА а2
    в
    ВВЕРХ
    ВПРАВО
    ВНИЗ
    ВНИЗ
КОНЕЦ
ПРОЦЕДУРА в
    ЦИКЛ 7 ВПРАВО
КОНЕЦ
```

Корректор

Среда обитания. Корректор — робот, подобно машине Тьюринга работающий с длинной лентой, разбитой на ячейки. Кроме ленты, у него есть ящик, в который он может копировать символ с ленты или записывать из него копию символа на ленту. Цель работы исполнителя — корректировка информации на ленте.

Стандартные процедуры.

ПИСАТЬα — записать символ α в обозреваемую ячейку ленты;

ЯЩИК+ — символ с ленты копируется в ящик;

ЯЩИК— — символ из ящика копируется на ленту;

ВЛЕВО — шаг исполнителя по ленте влево;

ВПРАВО — шаг исполнителя по ленте вправо;

ПЛЮС — замена символа на ленте следующим по порядку в заданном алфавите символом;

МИНУС — замена символа на ленте предыдущим в заданном алфавите символом;

ГОЛОС — подача исполнителем звукового сигнала.

Алфавит исполнителя.

^ 0 1 2 3 4 5 6 7 8 9 а б в г д е ж з и к л м н о п р с т у ф х ц ч ш щ ъ ы ь э ю я f g i j l h q r s u v w y z — + / * ( ) [ ] . , ! ? ; : ′ ″ ⌑ % @

Первым в алфавите стоит специальный символ пустого места ^. На ленте пустые места закрашиваются, в то время как символ пробела ␣ изображается чистой ячейкой.

Структуры. Вызов процедуры. В отличие от Таракана, где рекурсивный вызов непродуктивен, здесь рекурсия является основным методом программирования.

Развилка. Условием в развилке может служить любой символ алфавита Корректора. Оно считается выполненным, если в обозреваемой ячейке записан тот же символ. Символы $<$, $>$, $=$, $\neq$ не входят в алфавит исполнителя и служат для сравнения содержимого ящика и обозреваемой ячейки. Смысл этих знаков: символ в ящике соответственно предшествует, следует за, идентичен, не идентичен символу в обозреваемой ячейке в смысле алфавита Корректора.

Примеры задач для Корректора

1. Загнать мяч в корзину (рис. 5).

Р е ш е н и е.

```
ПРОЦЕДУРА мяч
    ЦИКЛ 5 шаг
КОНЕЦ
```

5 

```
ПРОЦЕДУРА шаг
    ПИСАТЬ␣
    ВПРАВО
    ПИСАТЬ*
КОНЕЦ
```

2 (модификация задачи 1). Между мячом и корзиной произвольное количество клеток.

Р е ш е н и е.

```
ПРОЦЕДУРА мяч1
    ПИСАТЬ␣
    ВПРАВО
    ЕСЛИ к ТО ПИСАТЬ*
      ИНАЧЕ е
    ГОЛОС
КОНЕЦ
ПРОЦЕДУРА е
    ПИСАТЬ *
    мяч1
КОНЕЦ
```

3. К десятичному числу на ленте прибавить единицу. В начальный момент Корректор стоит против младшей цифры числа. Пример возможной начальной ситуации на рис. 6.

6 

Р е ш е н и е:

```
ПРОЦЕДУРА плюс1
    ЕСЛИ^ТО ПИСАТЬ 1
      ИНАЧЕ п1
    ГОЛОС
КОНЕЦ
ПРОЦЕДУРА п1
    ЕСЛИ 9 ТО п2
      ИНАЧЕ ПЛЮС
КОНЕЦ
ПРОЦЕДУРА п2
    ПИСАТЬ 0
    ВЛЕВО
    плюс 1
КОНЕЦ
```

4. В тексте на ленте найти символ, имеющий максимальный порядковый номер в алфавите. В начальный момент исполнитель обозревает левый символ текста.

Р е ш е н и е.

Сначала поместим в ящик первый символ текста. Затем Корректор идет по тексту вправо, сравнивая содержимое ячейки на ленте с символом, находящимся в ящике. Всякий раз, когда

обнаруживается символ, имеющий больший порядковый номер по сравнению с символом из ящика, помещаем его в ящик.

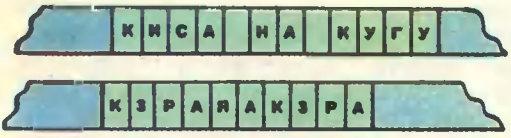
```
ПРОЦЕДУРА максимум
    ЯЩИК+
    х
    ГОЛОС
КОНЕЦ
ПРОЦЕДУРА х
    ВПРАВО
    ЕСЛИ < ТО ЯЩИК+
    ЕСЛИ ^ ИНАЧЕ х
КОНЕЦ
```

5. Какой текст будет записан на ленту после выполнения исполнителем программы:

а) «киса» (рис. 7);

б) «кзра» (рис. 8)?

```
ПРОЦЕДУРА киса
    ЦИКЛ 12 киса1
    ГОЛОС
КОНЕЦ
ПРОЦЕДУРА киса1
    ВПРАВО
    ЕСЛИ к ТО ПИСАТЬ л
КОНЕЦ
ПРОЦЕДУРА кзра
    ЦИКЛ 10 кзра1
    ГОЛОС
КОНЕЦ
ПРОЦЕДУРА кзра1
    ВПРАВО
    ЕСЛИ а ИНАЧЕ ПЛЮС
КОНЕЦ
```



7

8

Ответы:

а) лиса на лугу;

б) лиса алиса.

КРАВЦОВА А.
Москва

Работаем с микрокалькуляторами

Принцип наглядности в курсе информатики

«Задача повышения эффективности и качества обучения требует, чтобы все преподаватели переходили от интуитивного подбора приемов и средств обучения к научно обоснованному выбору оптимальных вариантов использования комплекса средств обучения» [2]. В полной мере это относится к преподаванию курса «Основы информатики и вычислительной техники», педагогическая платформа которого только еще устанавливается. Любой методический вопрос, возникающий у преподавателя этого курса, должен решаться с учетом того огромного педагогического опыта, который уже накоплен теорией педагогики применительно к другим дисциплинам. Когда перед нами встал вопрос о создании наглядных учебных пособий для курса «Основы информатики и вычислительной техники», мы прежде всего обратились к педагогической литературе. Описанное ниже наглядное учебное пособие разработано на основании теоретического подхода к роли наглядности в обучении, изложенного в работах Г. И. Хозяинова, Р. Г. Чураковой и З. С. Харьковского. Коротко о требованиях к наглядным учебным пособиям, выдвигаемых этими авторами.

1. Реализация принципа единства абстрактного и конкретного в познании с учетом особенностей учебного материала. Теоретические положения данного принципа базируются на основных закономерностях теории познания. Каждый учащийся как субъект познания создает у себя чувственно-наглядный образ изучаемого объекта, а затем, путем определенных мыслительных операций, образует ряд абстракций, создавая образ этого объекта. Уровень абстрактного мышления учащегося определяет скорость

образования адекватных представлений и, соответственно, скорость усвоения знаний. Если же у учащегося возникают трудности в образовании абстракций, то в учебном процессе необходимо использование таких наглядных учебных пособий, которые должны конкретизировать трудно создаваемый абстрактный образ.

2. Многофункциональность наглядного учебного пособия. «В структуре обучающей деятельности педагога различают три функции: информационную, когда он выступает в роли источника информации; функцию организации и управления учебным процессом; функцию личности обучаемого» [1]. Наглядное учебное пособие должно использоваться преподавателем при реализации всех этих функций. Структура наглядного пособия должна быть таковой, чтобы его можно было использовать и при объяснении нового материала и для проверки того, как усваиваются знания.

3. Использование наглядного пособия в сочетании с определенным текстом. При демонстрации учащимся наглядного учебного пособия происходит воздействие на их зрительное восприятие. Объясняя, преподаватель воздействует на их слуховое восприятие. Точно найденный словесный образ, точно найденная формулировка задачи в значительной мере влияют на эффективность усвоения знаний учащимися.

4. Изоморфизм и простота наглядного учебного пособия. Понятие изоморфизма не нуждается в пояснении — оно очевидно, а требование простоты заключается в том, «чтобы учащиеся могли охватить одним взглядом смысл — целостное содержание новой информации» [2].

5. Художественное оформление наглядного учебного пособия. Это требование является крайне важным, так как «наглядные средства обучения позволяют управлять вниманием, концентрировать его, перераспределять, поддерживать его устойчивость» [1].

Безусловно, все перечисленные требования должны быть использованы с учетом специфики данного учебного заведения, с учетом возраста учащихся, уровня их развития, задач обучения.

Учитывая все это, мы разработали макет для использования его в курсе «Основы информатики и вычислительной техники», который читается в техникуме с вечерней формой обучения на базе X классов. Объем курса 40 часов. Материально-техническая база — программируемые микрокалькуляторы «Электроника МК-56».

Макет, который мы используем на занятиях

Макет представляет собой три независимые панели, сделанные из листового алюминия толщиной 10 мм и окрашенные в светло-желтый цвет. Размер одной панели 560×380×10 мм. На лицевой стороне каждой панели выполнено 10 горизонтальных пазов, образующих наборное поле данной панели. Длина каждого паза 560 мм, глубина 2, внутренняя ширина 45, наружная ширина 40 мм. Пазы изготовлены таким образом, чтобы в них можно было вставлять с обеих сторон пластинки двух типов: квадратные из оргстекла и прямоугольные из алюминия, при этом пластинки должны удерживаться в пазах. С левой стороны по вертикали в каждом пазе просверлены отверстия диаметром 6 мм. Эти отверстия сквозные, для установки коммутаторных лампочек напряжением 12 В. С обратной стороны панели в металлическом корпусе закреплены 10 арматур для лампочек. Питание осуществляется от трех плоских батареек напряжением 4,5 В каждая. Для осуществления световой сигнализации панели предусмотрен пульт управления размером 160×50×40 мм, в который вмонтированы 10 микровыключателей. Они автономно включают и выключают каждую лампочку. Отдельно от панелей изготовлено два набора пластинок. Один — квадратные пластинки из оргстекла 45×45×1 мм. На каждой пластинке темно-зеленой краской по трафарету нанесены адреса ячеек программной памяти микрокалькулятора. Эти пластинки вставляются в пазы наборного поля на панели перед лампочками (одна пластинка в один паз). Другой набор представляет собой

пластинки из алюминия 320×45×1 мм. Пластинки окрашены в тот же светло-желтый цвет, что и панель. На одной пластинке черной краской по трафарету написана одна одноадресная или часть двухадресной команды и ее код. Эти пластинки вставляются в пазы наборного поля со стороны, противоположной той, с которой вставляются квадраты из оргстекла. Одна пластинка вставляется в один паз. Некоторые команды, такие, как команды организации цикла, условных переходов, написаны красным цветом. Таким образом, в одном пазе панели можно набрать одну одноадресную команду, а в двух пазах — одну двухадресную команду. На трех панелях можно набрать, соответственно, большее число команд. Изображение одной панели с набранным фрагментом программы представлено на рисунке. Кроме того, вверху панели находится ручка, с обратной стороны панели закреплен упор.

Так каким же образом изложенные выше педагогические требования к созданию наглядных учебных пособий реализуются в данном макете? Остановимся только на реализации первого принципа.

После первого года обучения курсу «Основы информатики и вычислительной техники» нам стало ясно, что сравнительно труден для наших учащихся вопрос о передвижении информации во внутренней памяти программируемого микрокалькулятора при выполнении программ, содержащих одновременно циклические конструкции, разветвления, подпрограммы. Учитывая общность этих вопросов при программировании на ЭВМ (обмен данными с программами), мы и разработали такой макет. Он представляет собой одну из материализованных форм логического представления организации внутренней памяти программируемого микрокалькулятора. Каждый паз моделирует одну ячейку памяти, наборное поле панели — блок ячеек памяти. Причем одну панель можно интерпретировать как блок программной памяти, другую — как блок стековой памяти, а третью — как блок регистровой памяти. Все зависит от того, какие педагогические задачи решаются с помощью этого макета.

Литература

1. *Хозяинов Г. И.* Некоторые гносеологические вопросы наглядности в обучении. М.: Знание, 1976.

2. *Чуракова Р. Г., Харьковский З. С.* Приемы и средства обучения. Вып. 3. М.: Знание, 1981.

ШОЛОМИЙ К.
Канд. психол. наук

Построение обучающей программы

Процесс компьютеризации обучения сопровождается серьезными трудностями, среди которых сегодняшний недостаток техники — далеко не главная. Вопрос технического обеспечения будет решаться, по-видимому, быстро. Более значительной проблемой является дефицит программного обеспечения. Ситуация здесь весьма серьезная, и думается, не исключает такой возможности, что поступление компьютерной техники в школы будет опережать создание программного обеспечения. Естественно, программы должны оцениваться с точки зрения своих дидактических возможностей. Необходимо также учитывать, насколько они совершенны в психолого-педагогическом отношении.

В программистской практике при организации диалога с компьютером широкое распространение получил так называемый «метод меню», известный в педагогике и психологии как задания с множественным выбором. Однако эти задания приучают к деятельности, которая, как правило, не находит применения при решении практических задач; препятствует развитию творческих способностей, поскольку ответ можно угадать. Недостатки «метода меню» предлагалось преодолевать с помощью более или менее удачных приемов[1], но совсем устранить их, по-видимому, не удастся.

Ниже описывается подход, при котором в качестве методологической основы берется модель знания, являющегося целью обучения.

Модель знания, являющегося целью обучения

В данной модели задачи рассматриваются как объекты некоторого класса, разбитого на подклассы по признакам, которые формально связаны с приводящими к решению операциями. Знание понимается как система эталонов, позволяющая фиксировать наличие у задач признаков, определяющих их принадлежность к тому или иному подклассу. Решение некоторой конкретной задачи рассматривается как распознавание, т. е. как установление принадлежности задачи по присущим ей признакам к подклассу, связанному с той или иной решающей операцией.

В качестве модели знания, удовлетворяющей такой интерпретации, принимается *дерево признаков*, т. е. граф типа «дерево», изображающий правило, которое обеспечивает решение множества задач данной учебной темы. Дерево признаков состоит, во-первых, из классификационного дерева, которое разбивает исходное множество задач на подклассы, связанные с решающими операциями. Во-вторых, в состав дерева признаков входит множество решающих операций, которые рассматриваются как используемые в процессе распознавания обозначения (наименования) подклассов.

На схеме 1 изображено дерево признаков, являющееся моделью знания по теме русского языка «Запятые при причастных и деепричастных оборотах». Приведем соответствующее правило.

Если причастный оборот стоит после определяемого слова, то он выделяется на письме запятыми и не выделяется, если стоит перед этим словом. На письме деепричастные обороты и одиночные деепричастия выделяются запятыми[2].

Исходным для данного дерева явля-

[1] Например, для предотвращения угадывания ответов было предложено значительно увеличить, до десятков и даже сотен, количество вариантов ответа. Это, несомненно, нерациональный прием. Превращение решения задачи в поиск иголки в стоге сена существенно увеличивает нагрузку на внимание, способствует быстрой утомляемости учащегося, засоряет его память.

[2] См.: *Баранов М. Т.* и др. Русский язык: Учебник для 5—6 классов. М., 1984. С. 228, 264.

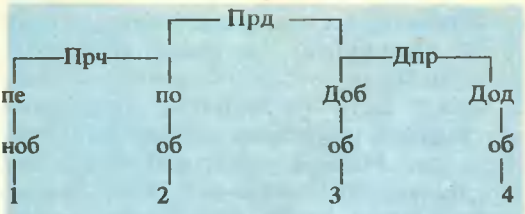


Схема 1. Модель знания по теме «Запятые при причастных и деепричастных оборотах».

Прд — предложение, входящее в состав исходного множества; Прч — в предложении есть причастие, ДПР — в предложении есть деепричастие, пе — причастие стоит перед определяемым словом, по — причастие стоит после определяемого слова, Доб — в предложении есть деепричастный оборот, Дод — в предложении есть деепричастие; об — обособляется, ноб — не обособляется; 1—4 номера подклассов.

ется множество всех возможных предложений, в которых требуется расставить запятые и где содержится либо причастный, либо деепричастный оборот, или одиночное деепричастие. Исходное множество разбивается классификационным деревом на четыре подкласса по следующим признакам: в предложении есть либо причастие, либо деепричастие; если причастие, то причастный оборот стоит либо перед определяемым словом, либо после него; если деепричастие, то оно либо одиночное, либо входит в состав деепричастного оборота. Признаки подклассов составляют ветви дерева. Например, первый подкласс характеризуется следующими признаками: в предложении есть причастие, причастный оборот стоит перед определяемым словом. Обозначениями подклассов служат символы операций обособления или необособления запятыми, которые соединены на графе штриховыми линиями с концевыми точками ветвей и определяют логическую связку «если..., то». Каждая ветвь выражает подправило: если предложение обладает перечисленными в ветви признаками, то содержащийся в нем грамматический элемент (причастный оборот, деепричастный и одиночное деепричастие) либо выделяется запятыми, либо не выделяется.

Данную модель можно применять и в таких далеких от грамматики предметах, как математика, физика, химия и т. п., где законы (теоремы) чаще всего даются имплицитно, т. е. выводятся из системы постулатов.

Например, выбор формулы можно рассматривать как распознавание принадлежности задачи к определенному классу задач, решаемых по этой формуле. Скажем, в физике задачи, решаемые по формуле $S=\frac{at^2}{2}$, можно охарактеризовать следующим правилом. *Задачи на прямолинейное равноускоренное движение без начальной скорости или прямолинейное равнозамедленное движение с равной нулю конечной скоростью, в которых требуется определить путь, пройденный движущимся телом, а также время или ускорение, решаются по формуле* $S=\frac{at^2}{2}$.

Выявив все правила, относящиеся к некоторой учебной теме, можно построить дерево признаков, определяющее решение задач данной темы, в котором обозначениями подклассов задач служат формулы. Применение этого метода к задачам по теме «Кинематика» ранее описано нами[3]. По аналогичной схеме можно строить модели, охватывающие несколько тем и целые учебные предметы.

Несколько слов об организационной стороне компьютерного обучения. Поскольку о полной передаче компьютеру обучающих функций не может быть и речи, возникает вопрос: какую часть учебного процесса следует доверить компьютеру? Попытаемся на него ответить.

Процесс обучения можно условно разделить на два этапа. На первом этапе учащийся должен запомнить необходимые сведения. Он может, например, запомнить определения понятий, формулировку правил, формулы и т. п., но он еще не умеет применять их на практике, т. е. при решении конкретных задач. На втором этапе (этап тренировки) учащийся выполняет ряд упражнений, решает задачи на основе приобретенных на первом этапе знаний, дово-

[3] См.: *Шоломий К. И.* Алгоритмизация выбора формул при решении физических задач // Проблемы программированного обучения физике и математике. Владимир, 1973.

дя эти знания до уровня устойчивых умений и навыков.

Эти этапы различаются прежде всего ролью учителя и уровнем самостоятельности действий учащегося. На первом этапе обучение осуществляется при максимальной помощи со стороны учителя. Эта помощь может варьировать по объему и содержанию (применение в обучении метода проб и ошибок, проблемного метода, метода П. Я. Гальперина и т. п.), но в любом случае учитель является здесь главным действующим лицом. Напротив, на этапе тренировки помощь учителя сводится к минимуму, а роль самостоятельных действий учащегося приобретает доминирующее значение.

Учитывая это, можно утверждать, что в настоящее время обучающий компьютерный диалог целесообразно применять преимущественно на этапе тренировки. Приведем некоторые аргументы в пользу этого утверждения. Личность учителя может играть на первом этапе огромную стимулирующую роль, для которой никакого эквивалента при компьютерном обучении не существует. На этапе тренировки, где должна преобладать самостоятельная работа учащегося, значимость личности учителя снижается. С помощью компьютера трудно обеспечить ответы на вопросы учащегося, возникающие на первом этапе обучения. Эта трудность сегодня непреодолима в силу непредсказуемости тематики вопросов и недоступности для компьютера диалога на уровне семантики. Лишение же учащихся возможности задавать вопросы ведет к угасанию интереса и снижению качества усвоения.

На этапе тренировки область возможных неясностей становится у́же и их разъяснение может осуществляться на более или менее формальном уровне, доступном реализации с помощью компьютера. Существенно, что при формализации этапа тренировки, в отличие от начального этапа запоминания, повышается вероятность достижения цели обучения. Такая формализация позволяет значительно увеличить количество решаемых учеником задач, расширяя возможности диагностики дефектов усвоения, а также их устранения.

Наконец, весьма существенно, что на этапе тренировки описанный выше подход позволит в значительной степени, а может быть и полностью, отказаться от заданий с множественным выбором. При составлении программ возможно предусмотреть, чтобы каждый подкласс набора имел свой обобщенный эталон (эталоны) правильного ответа, связанный с определенными реакциями машины. Так как компьютер в процессе работы всегда будет «знать», к какому подклассу принадлежит предъявленное упражнение, он тем самым будет «знать» и требуемые способы реагирования на ответы учащегося. В результате диалог приобретает однозначный характер.

Отметим, что программы, реализующие этап тренировки, можно применять и для контроля усвоения знаний. Главным компонентом такого контроля является проверка умения решать задачи данной учебной темы, т. е. задачи того типа, которые входят в состав программы, предназначенной для этапа тренировки. Поэтому, введя в программу некоторые дополнения, отражающие специфику контроля (невозможность подсказок, вызова правил, реагирование на ошибки и т. п.), ее можно использовать как программу двухцелевого назначения: и для тренировки, и для контроля усвоения.

В заключение приведем пример практической реализации предложенного подхода.

Программа для компьютерного обучения, составленная на основе модели знания

На основе рассмотренной выше модели знания в отделе информатики Московского областного института усовершенствования учителей под руководством автора статьи была составлена программа для компьютерного обучения русскому языку по теме «Запятые при причастных и деепричастных оборотах»[4].

[4] «Запятые при причастных и деепричастных оборотах». Программа для микрокомпьютера «Ямаха». Сценарий и алгоритм — К. М. Шоло-

Программа рассчитана на учащихся VI класса.

Программа состоит из следующих блоков.

1. Набор тренировочных знаний.
2. Правило.
3. Комментарии к ошибкам.
4. Процедурная часть программы:

4.1. Реализация учебного диалога.

4.2. Фиксирование результатов обучения.

4.3. Редактирование заданий.

Тренировочный набор состоит из 90 заданий, разбитых на два вида и шесть подклассов по 15 заданий в каждом. Признаковый состав набора показан на схеме 2. Задания первого вида — предложения, в которых требуется изменить порядок слов: поставить причастный оборот перед определяемым словом (подкласс 1) и после него (подкласс 2); затем расставить в измененном предложении запятые или указать, что запятые не ставятся. Например (подкласс 1): *Вода в пруду была покрыта* (1) *листьями* (2) *с* (3) *деревьев* (4) *опавшими* (5). Выполнение задания состоит, во-первых, во введении числа 15342, которое определяет следующий порядок слов в предложении: *Вода в пруду была покрыта опавшими с деревьев листьями.* Во-вторых, учащийся должен указать, что в этом предложении причастный оборот не обособляется.

Задания второго вида — предложения, в которых требуется расставить запятые или указать, что запятые не ставятся. Сюда относятся предложения, в которых причастный оборот стоит перед определяемым словом (подкласс 3) или после определяемого слова (подкласс 4); предложения с деепричастным оборотом (подкласс 5) и с одиночным деепричастием (подкласс 6).

В программе предусмотрена возможность ввода новых заданий путем замены имеющихся. При этом система подклассов и количественный состав набора должны оставаться неизменными. Правило, усвоение которого является целью программы, приведено выше. В комментариях к ошибкам указывается, как надо выполнить задание, и приводится соответствующее правило.

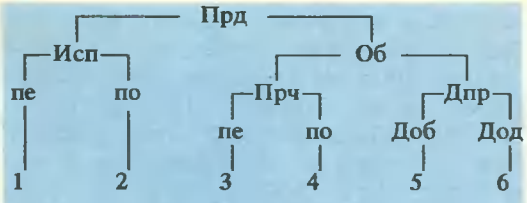


Схема 2. Признаковый состав набора тренировочных заданий по теме «Запятые при причастных и деепричастных оборотах». Исп — задания на изменение порядка слов и обособление; Об — задания только на обособление, 1—6 номера подклассов; другие обозначения см. схему 1.

Программа рассчитана на работу в двух режимах: тренировка и контроль. Для выбора режима перед началом работы на экран выводится соответствующая инструкция, в которой также отмечается возможность вызова правила после предъявления задания.

Выбрав режим, устанавливают время работы с каждым видом упражнений. Чтобы исключить какой-либо вид заданий, время для него задается равным нулю. При реализации диалога вначале даются предложения, в которых предлагается изменить порядок слов. При правильном ответе появляется слово «Правильно», которое сменяется предложением с исправленным, согласно ответу учащегося, порядком слов. Учащийся расставляет запятые или вводит сигнал о том, что запятые не ставятся. При адекватном ответе следует подтверждение и затем новое предложение. Если время истекло, машина переходит к заданиям второго вида. На экран выводится инструкция о расстановке запятых и задания подклассов 3, 4, 5 или 6. Если учащийся вводит «?», предъявляется правило, после чего восстанавливается исходная ситуация.

Иногда после правильной расстановки запятых машина предлагает ученику обосновать свой ответ (в среднем один раз на четыре предложения), предъявляя инструкцию: «Объясните расста-

мий, машинная реализация — Д. В. Лыхин, методическая консультация — В. В. Львов. М., 1986. Программа апробирована в Московском областном институте усовершенствования учителей. Она получила положительный отзыв учителей как русского языка и литературы, так и других специальностей.

новку или отсутствие запятых. Если забыли обозначения, введите «?». Ответом служит одно из следующих буквосочетаний: пе, по, Доб, Дод (см. схему 1). Если ответ правильный, машина печатает значение введенного буквосочетания, затем предъявляет следующее задание. При вводе «?» предъявляются перечисленные буквосочетания.

В случае неправильного ответа при изменении порядка слов и объяснении расстановки запятых предлагается подумать и попытаться ответить еще раз. Если ученик ошибается вторично, предлагается позвать учителя. Учитель вводит свой код, что вызывает появление исходного предложения, и дает требуемые разъяснения ученику. При неправильной расстановке запятых ошибка комментируется и подается новое задание, принадлежащее к тому же подклассу. При повторной ошибке машина отсылает к учителю.

В режиме контроля машина предъявляет задания и фиксирует ответы так же, как и в режиме тренировки, но не реагирует на ошибки и не выдает правила. Подсказка при объяснении расстановки запятых остается.

По истечении заданного времени поступает соответствующий сигнал вызвать учителя. Учитель вводит либо свой код, либо команду остановки. В первом случае устанавливается положение выбора режима, во втором — выдаются результаты работы учащегося:

Всего выполнено заданий . . . . .
Всего было ответов . . . . . .
Из них правильных . . . . . .
Ошибочных ответов . . . . . .
Ошибок в пунктуации . . . . . .
Ошибок в порядке слов . . . . . .
Ошибок при объяснении . . . . . .
Обращений к правилу . . . . . .

Заметим, что общее количество ответов может быть больше количества выполненных заданий, поскольку на одно задание дается до трех ответов: исправление предложений, расстановка запятых и объяснение расстановки запятых.

Блок редактора служит для коррекции введенных в машину предложений (внесения в них изменений, исправления положения на экране и т. п.). Он вызывается в положении «Выбор режима» вводом кода учителя.

ЧТО МОЖЕТ ЭВМ?

«Ритм» следит за ритмом

Испокон веков славился нефтяными богатствами Апшеронский полуостров — жемчужина Азербайджана. Но сейчас добывать нефть здесь стало все сложнее. Нефтяные скважины истощаются, и часть из них наполняется нефтью периодически, порою на несколько часов в сутки. Как же повысить производительность скважин?

В этом может помочь вычислительная техника — считают ученые Академии наук Азербайджанской ССР. Ими разработана новая система управления насосами «Ритм-3». Она следит за усилием, которое нужно приложить к штанге насоса. Как только оно станет меньше заданного (нефти в скважине нет), микрокомпьютер, анализируя создавшуюся ситуацию, отключает насос. Затем он снова заработает по заданной программе. Так осуществляется автоматическое регулирование процесса добычи нефти из скважин малой производительности.

Микросхема «Ритма» заключена в прочный герметический корпус, защищающий от перепадов температур и загрязнения брызгами нефти. Новая система проста в обслуживании и надежна в работе. Она не требует частых осмотров и ремонтов. Ее можно подключить к разветвленной сети автоматического управления оборудованием нескольких скважин.

Применение системы «Ритм-3» на нефтепромысле повышает производительность скважин, снижает расходы по ремонту и обслуживанию насосов.

Система программирования FOCAL (FOrmula CALculation) была разработана фирмой DEC в середине 60-х гг. и поставлялась с ЭВМ PDP-8. Впоследствии Фокал поставлялся в ОС RT-11 до четвертой версии включительно. Основное назначение Фокала — облегчение программирования внешних устройств в системах управления экспериментами, сбора и обработки данных. По сравнению с Бейсиком в Фокале более краткая запись команд. Их число существенно меньше, кроме того, имеются функции обращения непосредственно к адресам системной магистрали, что и позволяет управлять любыми внешними устройствами, к ней подключенными.

Фокал используется в БК-0010 из-за своей простоты, что особенно важно для микро-ЭВМ с малым объемом памяти. Мы называем Фокал системой программирования, потому что в ней для создания, отладки и выполнения программы пользователя применяются одни и те же средства.

Общая форма программы. Система программирования Фокал является интерпретирующей, т. е. все команды хранятся в памяти в виде текстовых строк. Система читает строки по порядку номеров и выполняет содержащиеся в них команды. Если мы вводим строку без номера, то система начинает выполнять ее сразу после нажатия клавиши *ввод*, в противном случае запоминает ее.

Интерпретирующей системе безразлично, в каком порядке введены строки. Это справедливо как для Бейсика, так и для Фокала.

Любая строка в программе должна иметь вид

НГ.НС ⟨пробел⟩ ⟨имя оператора⟩ ⟨пробел⟩ [⟨аргумент⟩, ⟨аргумент⟩...]

где НГ — номер группы строк, который может меняться от 1 до 99, НС — номер строки в группе, который может меняться от 01 до 99. Как уже было сказано, если номера группы и строки пропущены, то строка выполняется немедленно. Если пропущен только номер строки, то он считается равным 01.

Напишем простенькую программу, которая поможет нам понять основные принципы Фокала.

Пусть нам необходимо продемонстрировать выполнение пяти арифметических операций, доступных нам в Фокале, в порядке повышения их приоритета в арифметическом выражении. Для этого необходимо:

очистить экран;

принять с клавиатуры числа A и B;

вычислить результат операции с числами A и B и присвоить его значение переменной C;

вывести на экран формулу, числа и результат операции.

Программа выглядит следующим образом.

```
1.01 X FCHR (12);C ОЧИСТКА ЭКРАНА
1.05 T !,«ПРИМЕР ВЫПОЛНЕНИЯ
          ДОПУСТИМЫХ ОПЕРАЦИЙ»
1.10 A «ВВЕДИТЕ ЧИСЛО A»,A
1.11 A «ВВЕДИТЕ ЧИСЛО B»,B
2.01 C ВЫЧИСЛЕНИЕ
          РЕЗУЛЬТАТОВ ОПЕРАЦИЙ
2.10 S C=A-B
2.11 T !,"A-B=",A,"-",B,"=",C
2.20 S C=A+B
2.21 T !, "A+B=",A,"+",B,"=",C
2.30 S C=A*B
2.31 T !,"A*B=",A,"*",B,"="C
2.40 S C=A/B
2.41 T!,"A/B=",A,"/",B,"=",C
2.50 S C=A_B
2.51 T !,"A_B=",A,"_",B,"=",C
```

Так как все действия выполняются интерпретатором Фокала, то для запуска программы необходимо указать, на какой номер строки передать управление. Это делается командой G (от GOTO).

G 1.01 *ввод*

Если номер строки не указан, то программа начнет выполняться со строки с

минимальным номером. Если указан только номер группы, то программа начнет выполняться со строки с минимальным номером в этой группе.

После нажатия клавиши *ввод* экран очистился и на нем появились приглашения, в ответ на которые мы введем числа, например 3 и 2:

ВВЕДИТЕ ЧИСЛО А:3 *ввод*
ВВЕДИТЕ ЧИСЛО В:2 *ввод*

После этого на экране появилась таблица:

```
А—Б=3.0000—2.0000=1.0000
А—В=3.0000+2.0000=5.0000
А*В=3.0000*2.0000=6.0000
А/В=3.0000/2.0000=1.5000
А↑В=3.0000↑2.0000=9.0000
```


Разберем входящие в эту программу операторы.

Оператор X (выполнить) вызывает выполнение встроенной функции, которых в Фокале 21, из них 8 допускают выполнение без присваивания значений какой-либо переменной. Опишем их.

FRAN(1) — начальная установка генератора случайных чисел. После этого многократное обращение к нему (без аргумента) будет давать случайную последовательность чисел.

FCHR(X1,X2,...,XN) — вывод символьной информации. Позволяет выводить на экран символы X1,X2,..., заданные десятичным кодом, в том числе и управляющие. В нашем случае код 12 соответствует клавише СБР, очищающей экран.

FX (1, АДРЕС, ДАННЫЕ) — позволяет записывать данные по указанному адресу системной магистрали.

FSBR (НГ. НС, АРГУМЕНТ) — выполнение подпрограммы пользователя. Значение аргумента присваивается специальной переменной, &, после чего управление передается на указанный номер группы и строки НГ.НС. Для возврата в основную программу в подпрограмме должен стоять оператор R (RETURN — возврат). При возврате значение переменной & сохраняется и может быть использовано в основной программе.

FK(ПОЗИЦИЯ, СТРОКА) — установка курсора на экране в текстовом режиме на указанную строку и позицию в строке. Выполнение данной функции не изменяет координаты точки в графическом режиме.

FT(КОП,X,Y) — управление точкой с координатами X и Y на экране в графическом режиме. При КОП=1 точка зажигается, при КОП=0 — гаснет. Координаты точки запоминаются как текущие.

FV(КОП,XX,YY) — рисование / гашение отрезка, начинающегося от текущих координат точки и кончающегося указанными в качестве аргументов. Новые координаты становятся текущими. Смысл КОП тот же, что и в операторе FT(КОП,X,Y).

FP(КОР,МАСКА) — работа с портом ввода / вывода, который позволяет управлять устройствами пользователя. Для их подключения используют контакты на правом разъеме, напряжение на которых +0,5 В, если в соответствующем бите порта записан 0, и +2,4 В, если записана 1. КОП может принимать значения 1 или 2. В качестве маски может использоваться восьмеричное число от 0 до 177777 или переменная с десятичным значением от —32768 до +32767.

В первой строке нашей программы после точки с запятой стоит еще один оператор C(COMMENT — комментарий). Все, что следует за ним, воспринимается Фокалом как комментарий и при выполнении игнорируется. Текст комментария хранится в памяти, поэтому при всей желательности комментировать программу применять этот оператор следует экономно.

В строке 1.05 мы видим основной оператор для вывода текстовой информации на экран T(TYPE — печатать). Оператор выводит значения аргументов, разделенных запятыми, некоторые из которых — особые. Так, восклицательный знак вызывает перевод курсора на экране в начало следующей строки. При достижении курсором нижней

границы экрана все изображение поднимается на одну строку вверх.

Текстовая строка, заключенная в кавычки, без изменений (и без кавычек) выводится на экран, а символ ⌧ выводит на экран значения всех переменных программы.

Оператор A в строках 1.10 и 1.11 (от ASK — спрашивать) позволяет принять (ввести с клавиатуры) значения для списка переменных, указанных в качестве аргументов. Если в списке задана текстовая строка, заключенная в кавычки, то она выводится на экран. Имена переменных (в нашем случае A и B) могут быть любой длины и состоять из латинских букв и цифр, но различаются Фокалом только по первым двум символам. Имя переменной не должно начинаться с буквы F или с цифры.

Операторы в строках 2.10, 2.20, 2,30, 2.40, 2.50 занимаются только вычислениями. S (от SET — установить) устанавливает значение переменной C равным некоторому выражению.

В нашем случае программа закончилась только потому, что был исчерпан список строк. Однако иногда необходимо прекратить работу не в последней строке, а в середине текста программы. Для этого используется оператор Q (QUIT — оставлять, покидать).

Оператор повторения Фокала F (FOR — для)

10.01 F N=K0,M,K1; ⟨операторы для выполнения⟩ по-русски можно описать так:

для переменной N, значения которой изменяются от K0 с шагом M до K1, выполнить операторы в данной строке. В качестве K0, K1 и M может использоваться любое арифметическое выражение. Фокал прекратит выполнение группы операторов при N>K1; если шаг приращения M пропущен, то считается, что M=1.

Иногда в строке не хватает места для операторов, которые необходимо выполнить. В таких случаях применяется оператор D перехода к выполнению группы строк (DO — делать), в качестве аргумента которого используется номер группы строк, подлежащих выполнению. Выполнение группы операторов заканчивается либо по исчерпанию строк в группе, либо по оператору R (возврат).

Оператор условного перехода I

1.10 I⟨выражение⟩ A,B,C

передает управление в зависимости от значения выражения; если оно меньше нуля, управление передается на строку с номером A, равно нулю — с номером B, больше нуля — с номером C.

Фрагмент программы с условным переходом может выглядеть так:

```
1.01 A"ВВЕДИТЕ ЗНАЧЕНИЕ A",A
1.02 A "ВВЕДИТЕ ЗНАЧЕНИЕ B",B
1.10 I (A—B) 1.20, 1.30, 1.40
1.20 T !,"A=",A,"B=",B,"A
                        МЕНЬШЕ B"
1.21 G 1.01
1.30 T !,"A=B=",A,"A
                        РАВНО B"
1.31 G 1.01
1.40 T !,"A=",A,"B=",B"A
                        БОЛЬШЕ B"
1.41 G 1.01
```

Фокал позволяет непосредственно обрабатывать только числовую информацию, поэтому в нем нет средств описания типов данных. В Фокале могут быть использованы любые числа в диапазоне от 10^{-38} до 10^{38} со знаком + или —, записанные в виде десятичных дробей или в экспоненциальном формате. Все они округляются до шести значащих цифр.

Фокал позволяет использовать одно- и двухмерные массивы. Индекс массива может быть числом, переменной или арифметическим выражением. Пределы изменения индекса:

от —128 до +127 для двухмерного массива;

от —32768 до +32767 для одномерного.

Индексы заключаются в круглые скобки, а в случае двухмерного массива разделя-

ются запятой. Реальный размер массива ограничен объемом оперативной памяти и существенно меньше теоретически возможного.

Для просмотра текста программы служит команда W (от WRITE — писать)

W НГ.НС

Если НС не указан, то выводится указанная группа строк, если нет ни НС, ни НГ, то выводится весь текст програмы.

Для изменения отдельной, уже введенной строки применяется команда M (MODIFY — модифицировать), в качестве аргумента которой указываются НГ и НС. После ввода этой команды на экран будет выведена указанная строка, которую можно редактировать с помощью клавиш:

→ — сдвиг курсора вправо;
← — сдвиг курсора влево;
|→ — размыкание текста строки;
←| — смыкание текста строки;
←+ — удаление символа перед курсором.

В разомкнутое место строки можно вставлять символы обычным образом. После окончания редактирования нужно нажать клавишу *ввод*.

С помощью команды E (ERASE — удалить) можно удалить переменные программы. Если в качестве аргумента использовать символ T, то будет удален текст программы. Для удаления и текста программы, и переменных используется аргумент A(ALL — все). Если в качестве аргумента будут указаны НГ и НС, то будет удалена только эта строка.

Встроенные функции. Функция — это подпрограмма, возвращающая в качестве результата значение какой-либо функции заданного аргумента. Программист должен указать в программе, какой переменной необходимо присвоить это значение, например:

10.20 CAW=FSIN(3.14159/4)

Аргумент функции — выражение, стоящее в скобках справа от имени функции. Сами функции также могут входить в состав арифметических выражений. Оператор T заменяет присвоение значения функции переменной выводом этого значения на экран, например:

*T FCOS(0.5000)
0.8776

Все это относится к следующим функциям:

FSIN(X) — синус (аргумент в радианах);
FCOS(X) — косинус (аргумент в радианах);
FTAN(X) — тангенс (аргумент в радианах);
FASIN(X) — арксинус ($|X|\leqslant 1$);
FACOS(X) — арккосинус ($|X|\leqslant 1$);
FATAN(X) — арктангенс;
FLOG(X) — натуральный логарифм ($X>0$);
FEXP(X) — показательная функция;
FSGN(X) — знак числа (1, если $X>0$, 0, если $X\leqslant 0$);
FITR(X) — целая часть числа;
FABS(H) — абсолютная величина числа;
FRAN( ) — случайное число;
FSQT(X) — корень квадратный из числа.

О функции FCHR(—1) следует сказать особо. Она возвращает десятичное значение кода символа нажатой клавиши. С помощью следующей короткой программы можно выводить на экран коды клавиш:

1.1. T FCHR (—1); G 1.1

Для чтения содержимого памяти по непосредственно указанному адресу системной магистрали используется функция FX(КОП, АДРЕС, МАСКА). Если КОП=+1, то функция считывает содержимое указанного адреса без изменения, если КОП=0, то из ячейки памяти будут прочитаны только те разряды, одноименные биты которых в маске равны единице.

Таким же образом осуществляется работа с портом ввода / вывода при помощи функции FP(КОП,МАСКА). Если КОП=0, то функция возвращает значение регистра ввода в соответствии с указанной маской. Если КОП=3, то таким же образом читается регистр вывода.

Функция FSBR(НГ.НС,X) возвращает значение специальной переменной &, которое вычисляется в подпрограмме пользователя.

Поправка. В предыдущем выпуске «Справочного листка» на с. 80, 17-я строка снизу в правой колонке, вместо ВК следует читать WBK.

КАБИНЕТ ВТ

АРХАНГЕЛЬСКИЙ А.

Мир ЭВМ

3. Высоко сижу, далеко гляжу, или Исследование системы

Дальнейшее изложение будет вестись, как будто никакой документации по системе у нас нет. Так будет понятней, откуда что взялось, а главное, мы научимся разбираться в новой системе. Такой путь — не единственный, но достаточно эффективный. Важно запоминать не готовые решения, а методы их получения.

Единственное, что нам известно, язык клавиатурного монитора близок к английскому, поэтому основной документацией будут русско-английский и англо-русский словари.

Нам встретятся термины *пользователь* (программа пользователя) и *оператор*. Понятия эти весьма близки. Оператор становится пользователем, используя систему для выполнения прикладных программ; пользователь становится оператором, садясь за терминал.

Первое, что нам понадобится, машинное время, продукт, несмотря на свое название, весьма ощутимый, осязаемый и имеющий свое качество. Под машинным временем понимается возможность работать в вычислительной системе. Если для распечатки текста в две-три странички нужно полчаса заправлять бумагу или менять красящую ленту, то качество машинного времени плохое. Программное обеспечение, входящее в вычислительную систему, тоже влияет на качество машинного времени, а каким образом, мы узнаем на собственном опыте.

Вторая необходимая вещь — машинные носители информации. Начальник машинного зала нашего ВЦ — Александр Геннадьевич — выдал нам две дискеты диаметром 203 мм и сказал, что они подготовлены для работы.

Наконец, третье — необходима операционная система RT-11. Попросим кого-нибудь из программистов записать на дискету ее версию V05.00, однозадачный монитор RT11SJ.SYS, базовый вариант, поставляемый фирмой DEC, и минимальный набор программ, необходимый для работы. Но как все это запустить в работу?

Первые ДВК поставлялись только с накопителями ГМД-70 или ГМД-1012, и единственный способ загрузки состоял в том, что необходимо было набрать на клавиатуре терминала адрес программы аппаратной загрузки 173000 и команду G (от GO — идти); загрузка происходила только с нулевого дисковода. В последних выпусках ДВК появился дополнительный сервис. Для загрузки системы необходимо набрать символ D (для накопителей типа ГМД-70 или ГМД-7012) или X (для накопителей

Окончание. Начало см.: Информатика и образование. 1987. № 1, 2.

НГМД-6022) и номер накопителя, в который вставлена дискета. С «соткой» дело еще проще. На клавишах пульта нужно набрать число 173000 и нажать клавиши *занесение адреса* и *пуск*. На экране появится символ ¤.

В стандарте ASCII он называется dollar, в отечественной документации — «знак денежной единицы», а программисты часто называют его «солнышко», «бычок», «жучок» или «клоп».

После этого нужно набрать на терминале имя и номер накопителя, с которого будет загружаться система, и нажать **ВК**.

Но прежде чем начать работу, познакомимся с нашими терминалами. У VDT-52130 практически нет (в нашем случае) средств управления — нужно только нажать на клавишу ON LINE, чтобы загорелся индикатор «ON LINE» и терминал оказался подключенным к ЭВМ.

У 15ИЭ-00-013 есть целая управляющая строка. Для установки режима дисплея курсор необходимо переместить в управляющую строку нажатием клавиши **СДВ** и передвинуть по ней в нужное место клавишами ← и →. Каждую цифру, под которой может быть установлен курсор, можно изменить нажатием клавиши ↓ («стрелка вниз с точкой» на дополнительном поле клавиатуры — не путать с восклицательным знаком). После установки нужных режимов клавишу **СДВ** следует отжать. Мы пока не будем обсуждать все режимы дисплея, покажем только, какая управляющая строка должна быть на экране:

9600 0010 1110 0000 09 15 00

- время
- система команд № 2
- отсутствие звукового сигнала при нажатии клавиши
- отсутствие звукового сигнала в 72-й позиции курсора
- дополнительный режим
- скорость обмена информацией терминала с ЭВМ

В рабочем режиме должны быть нажаты клавиши **ЛИН**, **РЕД**, **ДУП**, остальные фиксируемые клавиши должны быть отжаты.

Итак, наберем на клавиатуре

@ 173000G

—

В накопителе защелкало, и на экране появилось первое сообщение:

RT11SJ (S) V05.00
?MON—F—FILE NOT FOUND
DK:STARTS.COM

Система чем-то недовольна, но чем? И что это за загадочный файл, который она не нашла?

Первое, что мы должны исследовать, внешние запоминающие устройства.

3.1. Домовая книга, или Справочник устройства файловой структуры

На экране в начале строки стоит точка — это признак того, что KMON находится в памяти и готов к приему команд.

Узнаем сперва, что находится на системной дискете, т. е. прочтем справочник тома. В русско-английском словаре мы найдем, что справочник по-английски — directory и, помня, что язык KMON очень близок к английскому, попробуем набрать это слово в качестве команды. Поразмыслив, мы добавили через пробел (разделитель) имя устройства, на котором стоит интересующий нас том (ведь система не может угадать все наши желания). Итак:

.DIRECTORY DX1:

Нажали **ВК**, в накопителе защелкало и на экране появилось

?DIR—F—ERROR READING DIRECTORY

Посмотрев в словарь, мы узнаем, что это фраза означает «ошибка чтения справочника». Неприятно, но один положительный результат все-таки есть: мы убедились, что можем пользоваться программой DIR — ведь именно она выдала сообщение об ошибке.

Не читается справочник носителя, установленного на устройство DX1:. А по-

чему? Например, дискета плохая или она вообще не вставлена в дисковод.

Дискета у нас одна, с нее загружалась система — значит, она хорошая. А вот правый дисковод, который называется DX1:, действительно пуст. Системная дискета вставлена в левый дисковод, именуемый DXO:. Попробуем прочитать справочник с устройства DXO:.

.DIRECTORY DXO:

Удача! На экране появилась таблица.

```
SWAP      .SYS     26P    13—JAN—84
RT11SJ    .SYS     73P    10—MAR—84
TT        .SYS      2P    10—MAR—84
DX        .SYS      3P    10—MAR—84
SL        .SYS     13P    10—MAR—84
LP        .SYS      2P    10—MAR—84
MX        .SYS     10P    10—MAR—84
NL        .SYS      2P    10—MAR—84
RK        .SYS      3R    10—MAR—84
MT        .SYS      9P    10—MAR—84
DP        .SYS      3P    10—MAR—84
LD        .SYS      8P    10—MAR—84
PIP       .SAV     29     06—FEB—84
DUP       .SAV     45     06—FEB—84
DIR       .SAV     19     06—FEB—84
DUMP      .SAV      9     06—FEB—84
16 FILES , 256 BLOCKS
230 FREE BLOCKS
```

На экране она разбита на два столбца, но в журнале здесь и далее печатается в один.

Это очень здорово, мы увидели знакомые имена файлов. Но сразу возникают некоторые вопросы: во-первых, около каждого файла (точнее, напротив его спецификации) стоит какая-то дата. Что она означает? По здравому смыслу это может быть дата либо создания файла, либо его копирования. Но файлы нам копировали сегодня, а в таблице стоит 1984 г., значит, это дата создания.

По некоторым (техническим) причинам в справочниках устройств с последовательным доступом, например магнитной ленты, указывается дата копирования файла.

Второй вопрос — можно ли найти соответствие между списком файлов, который мы получили, и порядком расположения файлов на носителе? Может быть, цифры, стоящие рядом с именами файлов, нам помогут. Начнем анализ. Указанные в конце итоговые цифры говорят нам, что наши 16 файлов занимают 256 блоков. Если мы просуммируем цифры, стоящие рядом с именами файлов, то получим также 256 блоков; значит, это — размеры файлов в блоках. Итак, данная форма справочника говорит только о наличии файлов на носителе, но не об их расположении. А как же быть?

Единственный выход — предположить, что перед нами не полная форма справочника, раз нет такой важной информации, как расположение файлов. Помня, что KMON любит разговаривать на английском и система позволяет делать нам некоторые ошибки, попробуем применить ключ /FULL (полный).

.DIRECTORY/FULL DX0:

Получим результат:

```
SWAP      .SYS     26P    13—JAN—84
RT11SJ    .SYS     73P    10—MAR—84
TT        .SYS      2P    10—MAR—84
DX        .SYS      3P    10—MAR—84
SL        .SYS     13P    10—MAR—84
LP        .SYS      2P    10—MAR—84
MX        .SYS     10P    10—MAR—84
NL        .SYS      2P    10—MAR—84
RK        .SYS      3P    10—MAR—84
MT        .SYS      9P    10—MAR—84
DP        .SYS      3P    10—MAR—84
LD        .SYS      8P    10—MAR—84
PIP       .SAV     29     06—FEB—84
DUP       .SAV     45     06—FEB—84
DIR       .SAV     19     06—FEB—84
DUMP      .SAV      9     06—FEB—84
<UNUSED> 230
16 FILES, 256 BLOCKS
230 FREE BLOCKS
```

Информации добавилось немного. По англо-русскому словарю определим, что unused означает «неиспользуемый», т. е. не занятые пользователем для файлов зоны на устройстве. Более того, итоговые цифры, которые сопровождали распечатку каталога всякий раз, показывают, что из 486. имеющихся в нас блоков 230. свободны.

Но мы не узнали главного — порядка расположения файлов. А что означает — знать порядок расположения файлов на носителе? Какая информация в дополнение к существующей позволила бы нам точно сказать, как они расположены? По всей видимости, это должна быть какая-нибудь монотонно меняющаяся величина, например, порядковые номера файлов.

В устройствах с последовательным доступом используются именно последовательные номера файлов, которые называются позицией.

Или это могут быть номера блоков, с которых начинаются файлы. Вот и попробуем в качестве ключа использовать слово blocks:

.DIRECTORY/BLOCKS DX0:

Мы достигли чего хотели:

```
SWAP      .SYS      26P  13—JAN—84    8
RT11ST    .SYS      73P  10—MAR—84   34
TT        .SYS       2P  10—MAR—84  107
DX        .SYS       3P  10—MAR—84  109
SL        .SYS      13P  10—MAR—84  112
LP        .SYS       2P  10—MAR—84  125
MX        .SYS      10P  10—MAR—84  127
NL        .SYS       2P  10—MAR—84  137
RK        .SYS       3P  10—MAR—84  139
MT        .SYS       9P  10—MAR—84  142
DP        .SYS       3P  10—MAR—84  151
LD        .SYS       8P  10—MAR—84  154
PIP       .SAV      29   06—FEB—84  162
DUP       .SAV      45   06—FEB—84  191
DIR       .SAV      19   06—FEB—84  236
DUMP      .SAV       9   06—FEB—84  255
16 FILES, 256 BLOCKS
230 FREE BLOCKS
```


В конце каждой записи стоит номер начального блока файла. Несложными расчетами мы можем убедиться в этом.

Однако мы говорили о такой важной характеристике, как идентификатор тома и имя владельца. Найдя в словаре слово volume — том, попытаемся использовать его в качестве ключа.

.DIRECTORY/VOLUME DXO:

И опять мы оказались правы:

```
VOLUME ID:          RT11SJ V5.00
OWNER      :        АРХАНГЕЛСКИЙ
SWAP      .SYS      26P  13—JAN-84
RT11SJ    .SYS      73P  10—MAR—84
TT        .SYS       2P  10—MAR—84
DX        .SYS       3P  10—MAR—84
SL        .SYS      13P  10—MAR—84
LP        .SVS       2P  10—MAR—84
MX        .SVS      10P  10—MAR—84
NL        .SYS       2P  10—MAR—84
RK        .SYS       3P  10—MAR—84
MT        .SYS       9P  10—MAR—84
DP        .SYS       3P  10—MAR—84
LD        .SYS       8P  10—MAR—84
PIP       .SAV      29   06—FEB—84
DUP       .SAV      45   06—FEB—84
DIR       .SAV      19   06—FEB—84
DUMP      .SAV       9   06—FEB—84
16 FILES, 256 BLOCKS
230 FREE BLOCKS
```

Мы убедились, что на данной дискете записана операционная система RT-11 V5.00 с однозначным монитором SJ и принадлежит дискета Архангельскому.

Далее рассматривается использование других ключей программы DIR; в журнальном варианте эта часть опущена. Поиграйте с системой сами!

3.2. Как веревочке ни виться, или Все ли мы попробовали

Даже при самой богатой фантазии трудно придумать, что еще можно сделать с каталогом. Но это только потому, что мы еще не имеем опыта работы с системой. У нас всего одна дискета, и то наполовину пустая, да и работаем мы только первый день. Поэтому дальше рассматриваются дополнительные возможности программы DIR, о которых мы не могли догадаться.

Во-первых, посмотрим суммарную структуру каталога; правда, мы пока в ней не сможем разобраться, но благодаря этой информации в дальнейшем нам станут понятны некоторые особенности файловой системы. Итак, ключ /SUMMARY:

.DIRECTORY/SUMMARY DX0:

Вот что мы увидели

```
29—JUL—86

18 FILES IN SEGMENT 1
 1 AVAILABLE SEGMENTS, 1 IN USE
18 FILE,S, 260 BLOCKS
226 FREE BLOCKS
```

Действительно, непонятно. Оказывается, в нашем распоряжении один сегмент, он используется и в нем 18 файлов. Знакомые итоговые цифры подтверждают количество файлов. Намотаем это себе на ус, чтобы вскоре вспомнить.

Мы не могли этого ожидать, но в некоторых случаях можно посмотреть справочник удаленных файлов. В некоторых, потому что не всегда удается увидеть достоверную информацию о них. Когда и что мы увидим, как в этом разобраться, мы поймем после изучения файловой системы, а пока сделаем для памяти распечатку на бумаге каталога удаленных файлов, используя ключ /DELETED — удалить.

.DIRECTORY/OCTAL/PRINT/DELETED DXO:

Попытка сделана не зря:

```
29—JUL—86 OCTAL
BOOCK .DPR 342 03—MAR—86 414
0 FILES, 0 BLOCKS
342 FREE BLOCKS
```

Оказывается, на дискете были еще какие-то файлы.

3.2.1. Некоторые подробности, или Ключи и сокращения для интерпретатора командной строки

Вот и все, что мы можем сделать с помощью программы DIR. Именно программы, а не команды. К команде DIRECTORY мы еще вернемся, когда будем изучать другие программы. В том и состоит главная обязанность клавиатурного монитора, чтобы вызвать нужную программу для выполнения заданной команды и сообщить ей, что нужно делать. Однако мы обещали рассказать, как можно выполнить работу клавиатурного монитора самим. Когда это может понадобиться? Например, когда на системном устройстве нет места, допустим, для программы DIR.

В этом случае сперва нужно запустить программу DIR с того устройства, где она имеется. Для этого используем команду RUN (бежать).

```
.RUN DX0:DIR.SAV
```

В ответ мы увидели звездочку.

А дальше надо подумать. Мы знаем, что если в начале строки стоит звездочка, то программа вызвала интерпретатор командной строки (CSI) и требуется ввести командную строку в формате CSI. Так как по умолчанию вывод справочника (каталога) происходит на терминал, то мы можем просто указать имя устройства, на котором стоит носитель, каталог которого мы хотим посмотреть:

```
*DX0:
```

Тот же эффект мы получим, если введем командную строку полностью:

```
*TT:=DX0:
```

Здесь вместо имени устройства для вывода информации TT: может стоять любое другое, например LP:, или полностью спецификация выходного файла.

Таким образом, то, что мы делали с помощью команды

```
.DIRECTORY/PRINT DX0:
```

можно сделать так:

```
*LP:=DX0:
```

А вместо команды

```
.DIRECTORY/OUTPUT:DX0:DX0FIL DX0:
```

использовать

```
*DX0:DX0FIL=DX0:
```

А как же быть с остальными ключами? Совсем просто, если обладаешь хорошей памятью или имеешь специальную табличку, которую мы приведем ниже. Ключ для CSI состоит из одного символа, с теми же аргументами, которые были в командах KMON, но сокращенными до трех символов. Например, вывод каталога с ключом /SORT:NAME можно осуществить так:

```
*DX0:/S:NAM
```

В программе DIR названия ключей также можно сокращать.

Ключ	Сокращение для KMON	Значение ключа CSI
PROTECTION	PRO	T
BLOCKS	BL	B
POSITION	PO	B
BRIEF	BR	F
FAST	FA	F
FULL	FU	E
PRINTER	PRI	—
TERMINAL	TE	—
OUTPUT	OU	—
SUMMAPY	SU	N
COLUMNS	C	C:DVAL
ORDER	OR	S:VAL
SORT	SO	S:VAL
EXCLUDE	EX	P
ALPHABETIZE	ALD	A
OCTAL	OC	0
NEWFILES	NE	D
DATE	DA	D:DVAL
SINCE	SI	J:DVAL
BEFORE	BEF	K:DVAL
FREE	FR	M
VOLUMEID	VO	V:VAL
DELETED	DEL	Q
REVERSE	REV	R
BEGIN	BEG	G
NOPROTECTION	NOPRO	U

Подчеркнем еще раз, что несколько ключей можно применять одновременно, если они не противоречат по смыслу друг другу.

Ну вот и все. Осталось прервать работу программы DIR и вернуться опять в KMON — для этого нужно одновременно нажать клавиши **СТР** и **С**.

На экране появилась точка. Для завершения работы в RT-11 никаких специальных действий не требуется. Нужно вынуть дискету, выключить накопители, затем выключить все остальные устройства — и можно немного отдохнуть.

Конец первой части.

КУПАВЦЕВ А.
Канд. пед. наук
МВТУ им. Н. Э. Баумана

Урок с калькулятором

Использование ЭВМ при решении учебных задач начинается с выбора и осмысления математической модели: алгебраического квадратного или дифференциального уравнений. Например, модель-система двух линейных алгебраических уравнений $A_1X+B_1Y=C_1$ и $A_2X+B_2Y=C_2$. Система двух линейных алгебраических уравнений широко используется при решении различных задач в школе и вузе. Умелое их сведение к той или иной математической модели относится к необходимым компонентам искусства решать задачи. Приведем несколько примеров, заимствованных из физики. Решение традиционных задач по кинематике прямолинейного равноускоренного движения материальной точки, в которых требуется определить координату и модуль скорости в определенный момент времени по заданным ускорению и начальной скорости, можно представить как решение системы двух линейных алгебраических уравнений, в каждом из которых имеется по одному равному нулю коэффициенту при другой неизвестной величине.

Важнейший этап решения учебной задачи с помощью ЭВМ — реализация математической модели в машинной программе, предъявляет определенные требования к используемым программам. Они должны иметь понятную для учащихся структуру.

Для уменьшения затрат времени на ввод программы в ЭВМ необходимо, чтобы предназначенные для использования на занятиях программы были короткими. Это достигается путем сокращения, упрощения программы. В качестве примера приведена такая программа решения системы двух уравнений в программе для микрокалькулятора «Электроника МК-56». Столбцы указывают соответственно адрес шага, нажимаемую клавишу и код операции. Вызов числа из регистра памяти обозначен буквами ИП, а ввод — буквой П.

00	ИП1	61	15	—	11
01	ИП5	65	16	ИП7	67
02	X	12	17	÷	13
03	ИП4	64	18	С/П	50
04	ИП2	62	19	ИП1	61
05	X	12	20	ИП6	66
06	—	11	21	X	12
07	П7	47	22	ИП3	63
08	С/П	50	23	ИП4	64
09	ИП3	63	24	X	12
10	ИП5	65	25	—	11
11	X	12	26	ИП7	67
12	ИП6	66	27	÷	13
13	ИП2	62	28	С/П	50
14	X	12			

Для сокращения общего числа шагов программы опущены машинная проверка единственности решения системы уравнений и запись значений искомых величин в регистры памяти микрокалькулятора. Вместо них после вычисления значений определителя системы и искомых величин вводится команда С/П, останавливающая дальнейшее выполнение программы и выводящая на индикатор калькулятора вычисленные значения. После первого такого останова на идикаторе микрокалькулятора высвечивается значение определителя системы, Учащийся убеждается, что определитель системы не равен нулю, и нажимает клавишу С/П. Продолжается выполнение следующих команд программы. При следующем останове на индикаторе микрокалькулятора высвечивается значение искомой величины X. Учащийся записывает ее и, нажимая на клавишу С/П, производит дальнейшие вычисления величины Y.

В некоторых случаях программу можно сократить также за счет комбинированного введения исходных данных, определяя значения тех или иных коэффициентов непосредственно из составленных для решения задачи уравнений.

Введение исходных данных в машинную программу представляет особую заботу для учителей. В обучении редко случается, чтобы в новой задаче уча-

щиеся в точности повторили бы решения старой, но с новыми числовыми данными. Введенную один раз в ЭВМ программу можно использовать для решения других задач (если решение последних основывается на той же математической модели), внося необходимые изменения в данные. Планируя определенную последовательность решения задач, преподаватель может свести такую коррекцию программ к минимуму.

Применение ЭВМ для решения задач освобождает учащихся от ручных вычислений, утомительных громоздких математических преобразований, отнимающих значительную часть времени, отведенного на решение задачи. С другой стороны, изящество решения задач на ЭВМ не должно лишать учащихся той части их деятельности, которая составляет содержание познавательного процесса по данной учебной дисциплине: применение усвоенных знаний, поиск и обоснование способа решения задачи. Поэтому в необходимых случаях ЭВМ привлекают на занятия для выполнения вспомогательных операций и расчетов, отдельных фрагментов решения задачи. Важно активное и осмысленное применение обучающих ЭВМ.

СЕНИН Ю.
Канд. физ.-мат. наук

Вам нужна обучающая система?

В вычислительном центре Университета дружбы народов им. Патриса Лумумбы создана и внедрена автоматизированная обучающая информационно-контролирующая система «Инструктор». Она предназначена для студентов, аспирантов и преподавателей всех специальностей, включая гуманитарные.

Центральное место в системе занимает программа SYSBAS. Эту программу можно использовать для создания курсов по самым разным дисциплинам — физике, электротехнике, геологии, сельскохозяйственным, бухгалтерскому делу и т. д. «Генерирование» курса производится в диалоговом режиме с помощью специальной программы CRTBAS.

«Инструктор» системно независим и может использоваться на машинах разных классов (СМ-4, ДВК и др.).

Расскажем о работе этой обучающей системы на примере заложенного в нее курса языка программирования Бейсик.

Он служит как для первоначального обучения программированию, так и для освоения Бейсика квалифицированными пользователями, владеющими другими языками высокого уровня. Для этого в «Инструктор» включены различные методические приемы, позволяющие сделать процесс обучения гибким и динамичным, индивидуализировать его, предлагать дополнительную информацию при неверном ответе на вопрос.

В системе содержится описание операторов языка Бейсик и упражнения, закрепляющие полученные знания. Для начала можно вызвать необязательный урок «Мое первое занятие на Бейсике», облегчающий освоение клавиатуры дисплея и некоторых простейших операторов языка. После каждого следующего урока можно проверять свои силы, выполняя упражнения, при этом специальная подсистема оценивает результаты.

Каждый неправильный ответ приносит штрафное очко. После первого ошибочного ответа дается возможность ознакомиться с дополнительной информацией, при второй неудачной попытке «Инструктор» заставит заново ознакомиться с основным текстом к данному уроку. В некоторых ключевых уроках «Инструктор» предлагает перед основными упражнениями ознакомиться с материалом в упрощенной форме и проделать более простые упражнения, не идущие в общий зачет (эта возможность предусмотрена специально для слабо

подготовленных пользователей).

При ответе «Инструктор» может дать подсказку, но это обходится в 0,5 штрафных очка.

Все ошибки, а также время, затраченное на урок, запоминаются машиной для последующего анализа преподавателем.

Желающие приобрести систему «Инструктор» могут обратиться по адресу: 117198, Москва, ул. Миклухо-Маклая, 4, ВЦУДН, Самойлову К. Е., Сенину Ю. Е.

Семинар-практикум

С 23 по 27 февраля 1987 г. в Переславле-Залесском состоялся первый Всесоюзный семинар-практикум по проблемам создания педагогических программных средств для вычислительной техники учебных заведений системы профтехобразования. Он был организован Государственным комитетом СССР по профессионально-техническому образованию и Академией наук СССР.

В работе семинара участвовали инженерно-педагогические работники опорных СПТУ, занятые разработкой ППС, ученые ИПС АН СССР, Свердловского инженерно-педагогического института, научных учреждений Госпрофобра СССР, ответственные работники Госпрофобра СССР и госпрофобров союзных республик.

На семинаре обсуждались следующие вопросы: технология создания ПСС; организация разработки, экспертизы и тиражирования ППС в системе профтехобразования; перспективы и опыт внедрения ЭВТ в учебно-воспитательный процесс учебных заведений профтехобразования.

Разработчиками были намечены перспективы создания ППС. Отечественные и зарубежные ППС, предназначенные для использования в общеобразовательной и профессиональной подготовке (для ЭВМ «Электроника БК-0010», ДВК-2М, «Электроника ДЗ-28», «Ямаха», «Правец-82»), демонстрировались на семинаре. Участники обменялись разработками.

Оживленная дискуссия сопровождала многие доклады.

Участники семинара посетили СПТУ-6 Переславля-Залесского и Институт программных систем АН СССР, где познакомились с основными направлениями научных исследований, проводимых институтом, с опытом совместной работы профтехучилища и института по внедрению вычислительной техники в учебный процесс, совершенствованию подготовки квалифицированных рабочих.

Такая форма работы очень эффективна, и участники семинара высказали пожелание регулярно планировать подобные мероприятия.

ДОБИН В.

Испытание практикой

— Сегодня у нас гости,— Любовь Ивановна Золотарева, преподаватель информатики алма-атинского СПТУ-15, улыбнулась.— Впрочем, мы уже привыкли. Эти ребята — из девятого училища...

Оказалось, что в ее кабинете строго по графику проводят занятия учащиеся семнадцати училищ Алма-Аты. Не каждое СПТУ города пока еще оснащено так, как пятнадцатое, где в достатке микрокалькуляторов, прекрасный дисплейный кабинет оборудован «Ямахами», где досконально разработана методика преподавания нового предмета. Потому и стало СПТУ-15 опорным. А это ко многому обязывает. В первую очередь к тому, что здесь, в пятнадцатом, должна проводиться разнообразная учебно-методическая работа, сюда едут за опытом, за рекомендациями, тут надеются получить практическую помощь.

Любовь Ивановна Золотарева помочь готова всегда. Хотя это, разумеется, и хлопотно, и отнимает массу времени. Ей радостно видеть, как благодарны коллеги, которые знакомятся с ее разработками, с раздаточным материалом, с продуманной до мельчайших подробностей методикой преподавания предмета, до сих пор для многих таящего в себе массу неожиданностей. Значит, ее опыт интересен другим.

Опыт преподавания предмета «Основы информатики и вычислительной техники», накопленный Л. И. Золотаревой, действительно любопытен. Ей удалось решить, пожалуй, одну из сложнейших проблем — сделать изучение информатики воспитывающим, профессионально направленным.

Самое важное, считает она, четко разработать общую структуру проведения урока, который должен включать в себя такие обязательные элементы, как повторение, изучение нового материала, закрепление, индивидуальная самостоятельная работа учащихся.

Урок она начинает с повторения предыдущих тем, не боится, если это надо, повторять одни и те же вопросы на нескольких уроках. Психологи утверждают: скорость усвоения теории у разных учащихся различна. Она это учитывает. При опросе пользуется разработанными ею карточками, подготовленными к каждому уроку. Такой опрос достаточно быстро вводит учащихся в деловой ритм занятия.

Затем следует проверка домашнего задания. Л. И. Золотарева выдает учащимся персональные домашние задания — по индивидуальным карточкам. Обычно это задание небольшое и не слишком сложное, но оно требует от ребят внимания и способности самостоятельно мыслить, делать умозаключения. В карточках — примеры с учетом профессиональной направленности, взятые из курса физики, математики и других предметов. В СПТУ-15 готовят строителей. Поэтому преподаватель старается подбирать задания таким обра-

зом, чтобы они пригодились учащимся в их дальнейшей работе в мастерских и на стройке.

Задания выполняются на самой обычной бытовой полиэтиленовой пленке шариковой ручкой. Когда учащийся на уроке выходит к доске, он накладывает пленку на кодоскоп и на экране всем видны результаты его домашней работы, т. е. каждое домашнее задание проверяет не один лишь преподаватель, а вся группа. Таких домашних заданий Л. И. Золотарева проверяет 5—6 за урок, столько же выставляет оценок.

Иногда на дом задаются самостоятельные работы, выполнение которых начато в классе. При проверке тетрадей за эти работы обязательно выставляются оценки. Если первые самостоятельные работы, как правило, выполняют не все учащиеся, то вскоре обязательная их проверка дисциплинирует ребят.

Важный момент урока — объяснение нового материала. Методика его достаточно гибка, она позволяет преподавателю перестраиваться по ходу урока, чтобы сделать новый материал максимально понятным и доступным. Л. И. Золотарева обязательно придерживается следующих правил. Не должна нарушаться логика изучения тем. Например, в учебнике А. Ершова в теме «Составная команда ветвления» программа на алгоритмическом языке указана в примере раньше, чем сам алгоритм. Неопытные преподаватели, стараясь буквально придерживаться учебника, так и дают эту тему учащимся. Возникает непонимание, так как ребятам трудно написать программу, не зная алгоритма. В теме «Составная команда повторения» в примерах вообще нет алгоритма, а сразу даются программы. У учащихся создается неверное впечатление, что алгоритм — сам по себе, а программа — сама по себе. Нарушается логика объяснения нового материала. Л. И. Золотарева сначала представляет алгоритм в виде схемы, а уж потом, исходя из данного алгоритма, пишет программу. Материал становится ясным и понятным.

Очень важно при изучении нового материала учитывать особенности психологии ребят, их культурный и общеобразовательный уровень. В методических указаниях по информатике есть такой пример: «Ученые собрались на симпозиум. После симпозиума они обменялись визитными карточками. Количество визитных карточек — *n*. Вопрос: сколько ученых принимали участие в симпозиуме?» Л. И. Золотарева заметила, что слова «симпозиум», «визитная карточка» непонятны многим учащимся, отвлекают их от решения задачи, требуют лишнего времени на ненужные объяснения. И она трансформирует задачу, насыщая ее понятными для ребят реалиями. В новом виде задача выглядит так: «Группа учащихся работает в мастерской. У каждого учащегося есть шаблон, по которому он делает столько деталей, сколько учащихся в группе кроме него. Пересчитав готовые детали, надо определить, сколько учащихся работало в мастерской». Такую задачу ребята решают гораздо быстрее.

Л. И. Золотарева считает: во время объяснения нового материала и вообще во время занятий все учащиеся обязаны вести конспекты и самостоятельно составлять так называемые терминологические словари. В конспект заносятся ключевые моменты объясняемого материала, то, что необходимо подчеркнуть, выделяется цветными карандашами. Это не только облегчает усвоение незнакомого материала, но и приучает учащихся к порядку, эстетически воспитывает их.

Все основные понятия по курсу «Основы информатики и вычислительной техники» записываются в терминологический словарь. Некоторые преподаватели используют для этой цели заднюю обложку конспекта. По мнению Л. И. Золотаревой, это не совсем удобно, так как на уроке словарь должен быть всегда перед глазами учащихся, только тогда он станет истинным подспорьем в работе. Ведение терминологических словарей очень полезно и по другой причине. Психологи доказали: учащийся максимально продуктивно способен усваивать новый материал не более 15 мин. Затем должна следовать пауза-разрядка или смена рода деятельности. Этому и служит заполнение словаря.

Любопытна предложенная Л. И. Золотаревой методика заполнения словаря.

Орфография высвечивается на экране, а сам текст диктуется при помощи магнитофона. Зачем нужен магнитофон? Многие учащиеся невнимательны. Они не очень усердно слушают педагога, зная, что он, если потребуется, еще и еще раз повторит текст. Магнитофон не попросишь повторить. Его приходится слушать внимательно. Кстати, озвучивают текст сами учащиеся. Это еще больше привлекает их внимание к тому, что они должны записать.

Особо необходимо сказать о том, какую роль на уроке отводит Л. И. Золотарева техническим средствам обучения, в том числе электронно-вычислительной технике. Многие педагоги считают, что уроки информатики можно целиком свести к работе на персональных компьютерах. Это мнение ошибочно. Л. И. Золотарева убеждена: компьютер — это такое же подспорье преподавателю на уроке, как и кодоскоп или другие технические средства обучения. Использовать его надо чисто функционально, не забывая о главенствующей роли на уроке преподавателя, который обязан уметь донести до учащихся новый материал, умело используя по мере необходимости самые разные ТСО: компьютер, кодоскоп, слайды, киноаппаратуру. Только так можно резко повысить КПД урока.

В то же время учащиеся обязаны на уроке учиться думать, формулировать свои мысли, предлагать свои решения той или иной задачи. Можно считать неудачным урок, на котором говорит лишь один педагог, а учащиеся безучастны к происходящему. Они обязаны проявлять себя на уроке, не бояться ошибок, обнаруживать самостоятельность мышления. В этом она видит конечную цель изучения курса информатики, а вовсе не в том, что ребята должны изучить ту или иную компьютерную технику. Ибо, придя на производство, молодые строители столкнутся с такими производственными и управленческими задачами, которые не удастся решить, не обладая способностью выбрать единственно верный алгоритм их решения и не умея составить программу действий.

Потому так много внимания уделяет педагог самостоятельной работе учащихся. Самое печальное, когда учащиеся внешне вроде бы усваивают курс информатики, но усваивают его бездумно, механически, слепо следуя указаниям и советам педагога. Между тем это встречается довольно часто. Столкнулась с подобным же и Любовь Ивановна Золотарева. Занятия на I курсе в основном сводятся к практическим работам. Первую половину года учатся писать программы на МК в режиме программирования. Хорошо усвоив материал I курса, программы составлять они научились довольно быстро. Казалось бы, все отлично. Однако однажды преподаватель решила провести эксперимент: написав задание на доске, она в программе допустила две умышленные ошибки. И что же? Лишь несколько учащихся заметили и исправили ошибки педагога, остальные же, ничуть не засомневавшись, прогнали ту программу, которая была написана на доске.

Проанализировав результаты эксперимента, преподаватель пересмотрела свою работу на II курсе, особое внимание обратив на развитие творческого мышления учащихся.

Этому служат и используемые педагогом индивидуальные самостоятельные работы. Почему именно индивидуальные? Опыт показывает: когда учащиеся знают, что их задания дублируются, то большая часть времени у них уходит либо на списывание, либо на сравнение полученных результатов.

В карточке — несколько заданий. Первое задание легкое, с ним способны справиться все учащиеся. А если ты выполнил первое, куда охотнее берешься за второе. Оно уже сложнее, с ним справится не каждый. Но каждый должен показать максимум того, на что он способен.

Чтобы выявить эти способности, четко определить уровень знаний учащихся, Л. И. Золотарева использует своеобразную методику оценки выполнения самостоятельных работ. Допустим, дано задание написать алгоритм, составить программу и просчитать на МК в автоматическом режиме. Каждый учащийся на своем микрокалькуляторе получает результат. Затем на экране высвечива-

ются два ответа: один из них — верный, другой — неправильный, если нарушен алгоритм. По команде педагога все учащиеся поднимают карточку с номером своего результата (1 или 2). У кого получился иной ответ, поднимают карточку с номером 0. Казалось бы, нулевую карточку поднимать стыдно, но при подсчете иногда не срабатывает МК, поэтому можно подумать, что виноват микрокалькулятор, а не сам учащийся. Это успокаивает ребят. Преподаватель поднимает контрольную карточку с правильным результатом, и те, кто задание выполнил верно, ставят себе в тетрадь один балл. Так же выполняется второе, третье задание. К концу самостоятельной работы подводятся итоги. Поднимают руки сначала те, кто не набрал баллов. Их работы преподаватель проверит отдельно — после урока. Двойки за это не ставятся. Набравшие один балл ставят себе за работу 3. За два балла — 4. За три балла — 5. Тут налицо не только быстрая оценка лучших и неуспевающих, но и большой воспитательный эффект. Учащимся доверили оценить свою работу. Ко II курсу ребята оценивают свои знания вполне объективно.

Контроль знаний учащихся можно осуществлять и с помощью микрокалькулятора. В этих целях Л. И. Золотарева использует программу, разработанную преподавателем информатики В. В. Рейтером из ПТУ-5 г. Семипалатинска. Достоинство этой программы — в ее компактности и простоте.

Широкие возможности для проявления творчества предоставляются преподавателю при закреплении пройденного материала. В частности, на этом этапе урока легче всего достичь профессиональной направленности предмета.

Например, Л. И. Золотарева предлагает учащимся составить алгоритм изготовления или сварки цепи из колец. На микрокалькуляторе проводятся расчеты изготовления различных строительных деталей и элементов. Например, стамеска имеет определенный угол среза и определенную высоту. Требуется определить, на каком расстоянии от конца стамески надо сделать срез. Другое задание: рассчитать угол заточки сверл, изготовленных из разных материалов. Составляется общий алгоритм расчета заточки сверла, а затем учащиеся пишут программу на МК. Составляются программы решения и других экономических и производственных задач. Это приучает ребят к мысли о том, что они должны будут применять знания информатики на производстве. А до тех пор они стараются использовать их во время работы в мастерских. К примеру, делая молоток, учащиеся изготовляют сначала боек, а потом ручку — не потому, что им это подсказал мастер, а потому, что подобную задачу они решали на уроке информатике и убедились, что такой порядок изготовления инструмента позволяет экономить материал, время, рационально выполнять задание.

Привлекает не только методика преподавания предмета, разработанная Л. И. Золотаревой, но и оборудованный при ее активном участии кабинет информатики. Он, образно говоря, размещен в трех помещениях. В первом — учебный кабинет, где проходят теоретические занятия. Кабинет выразительно оформлен наглядной информацией, здесь оборудованы стенды, планшеты по курсу информатики. В столы, в специальные ящики, встроены микрокалькуляторы.

В соседнем кабинете оборудована учебная лаборатория — дисплейный класс. В особом помещении размещена ЭВМ, где также проводятся практические занятия. Таким образом, в СПТУ-15 создан комплекс аппаратуры и учебных кабинетов, позволяющий училищу вести разнообразную учебно-методическую работу.

Она проводится по нескольким направлениям. Во-первых, в этом опорном училище проходят обучения и повышение квалификации инженерно-педагогические работники местных училищ. В частности, здесь проведены занятия по ознакомлению с программируемыми калькуляторами и другой электронно-вычислительной техникой директоров СПТУ, их заместителей по учебно-производственной работе, преподавателей информатики, методистов, мастеров производственного обучения, работников областных управлений профтехобразования.

Здесь же, в СПТУ-15, проводятся открытые уроки информатики, на базе

опорного кабинета проведены творческие методические беседы и заседания городской методической секции преподавателей информатики.

Наконец, в училище постоянно проводятся экскурсии для школьников Алма-Аты, для студентов, готовящихся стать преподавателями спецпредметов, и мастеров производственного обучения.

Вся эта большая и разносторонняя учебно-методическая работа четко спланирована, что значительно повышает ее эффективность.

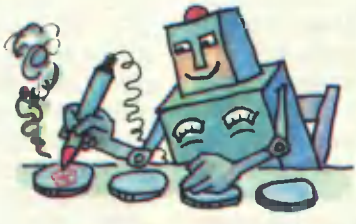
Однако этим не ограничивается работа опорного кабинета информатики. Под руководством Л. И. Золотаревой в училище создан и весьма интересно работает кружок информатики, где ребята не только еще основательнее изучают вычислительную технику, но и сами занимаются ее изготовлением. К примеру, они изготавливают демонстрационные микрокалькуляторы, выполняют заказы промышленных предприятий по составлению и расчету программ. Их работы отмечены многими наградами.

— Очень важно для педагога,— говорит Л. И. Золотарева,— чтобы изучение информатики не превратилось для ребят в самоцель. Они должны понимать, как можно применить полученные на практике знания. Как были довольны они, когда проведенные ими расчеты водоводных теплообменников и программы, составленные по заданию «Гидропроекта», получили высокую оценку специалистов. Ребята убедились: в своем кружке они делают важное и очень нужное производству дело.

ЧТО МОЖЕТ ЭВМ?

Лазер маркирует детали

Лазеры не новинка. Они сейчас широко применяются в промышленности. Но лазерная установка «Квант», созданная советскими инженерами, выгодно отличается от обычных квантовых генераторов. Ее луч по команде ЭВМ вычерчивает буквы, цифры, знаки на металле, пластмассе, стекле.



Новая установка незаменима при маркировке различной продукции, например подшипников, инструментов, транзисторов. Обычно это делается с помощью металлических штампов. Причем каждый из них пригоден лишь для определенного вида изделий. Поменялась их номенклатура, надо менять и маркировочные штампы. С помощью лазера выполняется маркировка любого твердого материала. При этом на экране дисплея можно изменять сочетание цифр и букв. ЭВМ запомнит этот текст и, управляя лазером, напишет его на маркируемой детали. Надписи можно делать на малых и сравнительно больших поверхностях. При этом лазерный луч не деформирует их как металлический штамп. Это позволяет маркировать детали сложной конфигурации.

ЭВМ в народном хозяйстве

БАРАНОВ Л., БУДАРИН В., ГОЛОВИЧЕР Я., СУХИН А.
Московский институт инженеров железнодорожного транспорта

Электроника в поезде

Машинист — одна из самых массовых и в то же время очень ответственная профессия. От профессиональных и гражданских качеств машиниста зависит надежность работы всего транспортного конвейера. Однако для облегчения и интенсификации его труда сделано пока еще очень мало. По существу, последними крупными достижениями в этой области были замена паровозной тяги на электрическую и дизельную, что существенно улучшило условия труда локомотивных бригад, и внедрение автоматической локомотивной сигнализации, позволяющей дублировать в кабине показания светофоров и обеспечивающей (да и то не во всех случаях) экстренную остановку поезда при проезде запрещающего сигнала. Тем временем заметно возросли интенсивность движения, массы поездов, а следовательно, и нервные нагрузки машинистов. Управляя, к примеру, мощным грузовым электровозом, нужно так выбирать режимы тяги и торможения, чтобы добиться минимального расхода электроэнергии, предотвратить боксование и юз, не превысить допустимых продольных усилий в составе (они могут и разорвать поезд!). Одновременно надо обеспечить безопасность движения (т. е. постоянно следить за сигналами светофоров, ограничениями скорости, состоянием пути) и контролировать работу оборудования электровоза. У машинистов пассажирских поездов свои проблемы — высокая скорость движения, необходимость плавного хода. На пригородных поездах и особенно в метрополитене добавляются частые остановки, которые надо выполнять с высокой точностью и за минимальное время (иначе не уложиться в график).

Естественно желание переложить хотя бы часть этих забот на электронику, однако решить такую задачу не просто. Машинист ориентируется не только на показания приборов, но и на многочисленные внешние факторы, использует опыт предыдущих поездок. Очевидно, что алгоритмы, с помощью которых можно пытаться автоматизировать эти функции, должны быть достаточно сложными.

Простейшие принципы автоматического управления, используемые, например, в станках с ЧПУ, для поезда зачастую неэффективны — ведь любые определенные заранее (путем расчетов либо по итогам опытных поездок) режимы управления могут оказаться нерациональными в данной конкретной ситуации и даже недопустимыми из-за действия внешних возмущений. К таким возмущениям относится не только изменение параметров движения (влияние ветра, изменение напряжения в контактной сети и т. д.), но и внеплановые ограничения скорости, связанные, например, с путевыми работами, а иногда и корова на рельсах. Для их учета зачастую целесообразно не просто кор-

ректировать выбранные заранее режимы управления, а заново выполнить расчет режима ведения поезда исходя из сложившейся ситуации. Реализация же алгоритмов, адаптирующихся к условиям движения, обеспечивающих оптимизацию расхода электроэнергии, невозможна без использования ЭВМ.

Первые попытки использования вычислительных машин на электроподвижном составе были предприняты в нашей стране в конце 50-х гг. Системы автоматического управления с бортовой ЭВМ испытывались на поездах пригородного сообщения и метрополитена. Однако уровень развития ЭВТ того времени не позволил перейти от экспериментальных образцов к промышленным. В течение следующих 15 лет системы автоведения строились по так называемой жесткой логике. Об использовании ЭВМ вновь заговорили лишь с появлением микропроцессора.

Каким же выглядит современный локомотив с микро-ЭВМ на борту? Это иерархическая, т. е. многоуровневая, система, каждый элемент которой представляет собой микропроцессорный контроллер, ориентированный на выполнение определенной задачи. Контроллеры низшего уровня управляют отдельными агрегатами, например каждым тяговым двигателем; на более высоких уровнях они проверяют работоспособность агрегатов и систем, обеспечивают безопасность движения поезда и, наконец, автоматизируют функции машиниста по управлению движением.

Бортовая система автоведения может работать автономно, выполняя заложенный в ее память график движения поезда, а может являться составной частью централизованной системы управления движением. В последнем случае ЭВМ центрального поста (самый верхний иерархический уровень), анализируя ситуацию на линии и график движения, определяет и передает на поезда значения времен хода по перегонам, команды на отправление со станций. Централизованные системы обладают значительно большими возможностями, облегчают работу диспетчеров и позволяют оптимизировать не только движение отдельного поезда, но и весь поток поездов в целом.

Наиболее проста структура централизованной системы автоведения поездов метрополитена, поскольку там длина линий невелика, все поезда однотипны и двигаются, как правило, с постоянным межпоездным интервалом. Она имеет трехуровневую структуру. Общее руководство ситуацией на линии осуществляет центральный пост управления, куда входит мощная ЭВМ, хранящая в памяти плановый график движения. Сюда со всех станций по специальным кабельным линиям стекается информация о прибытии и отправлении поездов. После ее анализа определяются необходимые корректировки времен стоянки и хода по перегону каждого поезда. Если по каким-либо причинам график движения нарушается (например, поезда начинают опаздывать с прибытием на станцию), специальная программа в ЭВМ центрального поста формирует параметры корректировки графика движения в целом, а затем они передаются в станционные устройства автоведения, основная функция которых — обмен данными между центральным постом и устройством автоведения поезда. Помимо этого, станционные устройства могут управлять информационными табло для пассажиров, контролировать прохождение поездами станций, подсчитывать входящих на станцию пассажиров и т. п. Связь станции с прибывающими поездами обеспечивает приемо-передающая аппаратура, работающая в инфракрасном диапазоне электромагнитных волн.

Станционные и поездные устройства имеют в своем составе универсальные микроконтроллеры. Именно они обеспечивают решение задач обмена информацией и управления.

Что представляет собой микроконтроллер? По сути, это обычная ЭВМ с процессором, памятью, устройствами ввода-вывода. От универсальной ЭВМ он отличается наличием прежде всего специализированных устройств сопряжения с объектом управления (интерфейсов). Они нужны устройствам автоведения для управления станционной и поездной аппаратурой и обработки информации, представленной сигналами, величина ко-

торых существенно отличается от принятых в вычислительной технике стандартов. Например, для управления движением поезда на перегоне, когда отсутствует непосредственная связь с центральным постом управления, необходимо знать положение поезда и скорость движения. Из существующих способов определения этих параметров на железнодорожном транспорте наибольшее распространение получила обработка сигнала индуктивного датчика, устанавливаемого на оси колесной пары; амплитуда и частота его сигнала пропорциональны скорости движения. Микроконтроллер же обрабатывает цифровую информацию, представленную в виде байтов или 16-разрядных слов, поэтому одна из функций интерфейса поездного устройства — преобразование аналогового сигнала осевого датчика в двоичный код.

Микроконтроллер должен также иметь информацию о состоянии узлов автоматики, схемы управления поездом, другой аппаратуры. Хотя она имеет двоичный вид («работает» — «не работает», что соответствует логическим «0» и «1»), непосредственно обрабатывать эти сигналы невозможно: они имеют уровни напряжения вагонной аккумуляторной батареи (75 В), а в сумме с импульсными помехами достигают амплитуды в несколько сотен вольт. Интерфейс ввода сигналов осуществляет фильтрацию и преобразование уровней, а также обеспечивает гальваническую развязку цепей управления поездом и микроконтроллера. Аналогичные функции выполняет и интерфейс вывода команд управления.

Особую задачу решают адаптеры канала связи. Они входят в состав станционного и поездного устройств и осуществляют последовательно-параллельное преобразование информации: технические средства инфракрасного канала связи предполагают наличие одного излучающего и одного приемного элемента и передают биты информации один за другим (последовательный код), а микроконтроллер работает в параллельных кодах (воспринимает информацию группами по 16 битов).

Теперь пора перейти к алгоритмам и программному обеспечению поездного устройства, непосредственно управляющего движением, но прежде скажем несколько слов о средствах, с помощью которых машинист управляет поездом.

Эксплуатируемые сейчас в метрополитене электропоезда оборудованы устройством дискретного управления, которое позволяет вести поезд в режимах реостатного контроллера, тяги, торможения и выбега. Изменение тягового тока обеспечивается включением двигателей (последовательно или попарно-параллельно), а также ослаблением поля обмотки возбуждения электродвигателя. В режиме торможения тяговые двигатели переключаются в генераторный режим, регулирование тока при этом производится изменением нагрузочного сопротивления в цепях якорей с помощью реостатного контроллера.

При ведении поезда машинист ориентируется на показания скоростемера, расстановку в тоннеле знаков включения (выключения) тяги и торможения, а также учитывает загрузку и динамические характеристики подвижного состава. Качество управления — точность выполнения графика, плавность хода и остановки поезда — полностью зависит от квалификации и опыта машиниста. В условиях высокой интенсивности движения субъективность восприятия и оценки машинистом текущей обстановки неизбежно ведет к нарушению графика и перерасходу электроэнергии.

Алгоритмы, заложенные в виде программ в память поездной микро-ЭВМ, реализуют законы управления, основываясь на точном знании текущей скорости и положения поезда на перегоне. В алгоритмах формализованы, т. е. представлены в виде формул и логических соотношений, опыт ручного управления и результаты моделирования на ЭВМ (проведенного для нахождения наиболее экономных режимов) процесса движения. В качестве примера рассмотрим наиболее сложный из них, управляющий торможением.

Исходной информацией для алгоритма являются координаты ограничений скорости, точек изменения профиля пути и коэффициенты усредненных тормозных характеристик данного типа подвижного состава. Он определяет рас-

согласование между фактической скоростью движения и текущей программной скоростью. По его величине рассогласования определяется режим торможения, при этом учитываются характеристики тормозных средств поезда, состояние схемы управления и скорость изменения величины рассогласования. Важно отметить, что реостатный контроллер имеет возможность только увеличивать эффективность торможения, так как в режимах тяги и торможения не имеет обратного хода. Это свойство схемы управления поездом накладывает дополнительные требования на точность определения режима управления.

Программное обеспечение бортового микроконтроллера — набор программ, реализующих алгоритмы управления поездом, обслуживания канала связи и интерфейсных блоков, тестирования технических средств поездного устройства. Инициирование (запуск) той из них, которая нужна в данный момент, осуществляет «главная» программа — диспетчер. Она на основе информации, поступающей в микроконтроллер через интерфейсные блоки, управляет работой всего программного обеспечения.

Все программы написаны на ассемблере. Этот язык неудобен для программиста, зато позволяет создавать наиболее эффективные программы, что в данном случае является самым важным обстоятельством.

Системы автоведения на магистральных железных дорогах имеют свою специфику. Прежде всего следует отметить, что структура централизованных систем и используемые в них алгоритмы должны быть намного сложнее, чем в метрополитене. Ведь здесь необходимо управлять поездами различных типов (грузовыми, пассажирскими, пригородными), при этом возрастает количество и сложность возможных ситуаций: обгоны одних поездов другими, принятие и отправление поездов одновременно на несколько направлений и т. д. Из-за большей протяженности линий, числа поездов и путей на станциях нужны более сложные каналы связи. Поэтому на первом этапе автоматизации на магистральных дорогах следует использовать автономные системы автоведения. Так поступили, например, железнодорожники Берлина. У нас в стране первые автономные микропроцессорные системы автоведения установлены на скоростных пассажирских электровозах ЧС200, развивающих скорость до 200 км/ч (они эксплуатируются на линии Ленинград—Москва). Кроме того, уже в этом году опытная партия микропроцессорных систем автоведения будет установлена на электропоездах Горьковского направления на Московской железной дороге. Эти устройства предназначены для автоматизации управления движением поезда по перегону и обеспечивают выполнение графика движения с точностью ± 30 с с одновременной оптимизацией расхода электроэнергии.

Устройство автономной системы автоведения во многом напоминает микропроцессорное поездное устройство в метрополитене. Это естественно, поскольку использующая средства вычислительной техники структура систем управления является универсальной, а настройка системы на решение задачи достигается в основном с помощью соответствующих интерфейсов и программного обеспечения.

Особенностью автономной системы автоведения является наличие специального блока — репрограммируемого постоянного запоминающего устройства, в котором хранятся исходные данные, описывающие маршруты движения поездов и их графики. Эта информация записывается заранее в локомотивном депо. Конструкция блока обеспечивает быструю установку его в систему, а также возможность быстрой замены на другой при оперативном внесении изменений в исходные данные.

Как же работает система автоведения, например, на пассажирском поезде? Перед началом поездки ЭВМ запрашивает номер поезда и станцию отправления. Их вводит машинист с помощью клавиатуры. Затем машина находит и считывает из блока хранения исходных данных информацию о предстоящем маршруте: график движения, значения и координаты подъемов и спусков, расстановку участков с ограниченной скоростью движения и т. д. Машинист может опе-

ративно корректировать некоторые исходные данные, например по указанию поездного диспетчера, переданному по радио, изменять время хода по некоторым перегонам. После этого управление передается системе автоведения. Исходя из параметров перегона, локомотива и вагонов, ЭВМ рассчитывает последовательность режимов управления поездом, обеспечивающую проезд до следующей станции точно за предусмотренное графиком время при минимальном расходе электроэнергии или топлива. Такими режимами могут быть, например, разгон с максимально допустимой по мощности локомотива и условиям сцепления колеса с рельсом силой тяги, движение с постоянной скоростью и т. д. Реализуются они также автоматически с помощью специальных программ. В качестве исходной информации при этом используются координата, скорость и ускорение поезда, измеряемые с помощью программы обработки импульсов, приходящих от осевого датчика. Другие программы производят опрос датчиков тока тяговых двигателей, давления в тормозной системе и т. д. Помимо программ расчета режимов управления, автоматического регулирования отдельных режимов и измерения текущих параметров программное обеспечение включает набор сервисных программ, обеспечивающих обмен оперативной информацией с машинистом и контроль работоспособности отдельных блоков.

Программа расчета режимов движения является фоновой, т. е. выполняется микропроцессором постоянно. Четыре раза в секунду ее прерывает таймер; в это время вызываются программы измерения и программы-регуляторы (скорости), реализующие выбранный до этого режим управления поездом. В процессе работы они могут, в свою очередь, запускать другие, в том числе и сервисные. Прерывание в работе программ вызывается не только таймером, но и сигналами некоторых датчиков, требующими немедленной реакции. По прерыванию запускается также обработка команд, поданных машинистом с пульта управления.

Разработке программного обеспечения системы автоведения предшествует этап создания алгоритмов управления, обеспечивающих реализацию автоматизируемых процессов с заданными показателями качества. Проблемы при этом возникают непростые. Рассмотрим, например, задачу оперативного расчета режимов управления поездом. Прежде всего следует отметить, что поскольку исходные данные для него могут меняться в процессе движения поезда по перегону, то такой расчет целесообразно не только проводить перед въездом на перегон, но и циклически повторять во время движения. Это позволит, например, корректировать перегонные времена по указаниям поездного диспетчера, оптимально реагировать на незапланированные снижения скорости и т. д. В таких случаях говорят, что ЭВМ должна работать в режиме реального времени, т. е. выполнять расчет непосредственно в момент принятия решения, исходя из измеренных значений параметров управляемого процесса. Очевидно, что скорость ее работы должна быть значительно выше скорости самого процесса, иначе принятые решения будут безнадежно опаздывать и окажутся неэффективными. С другой стороны, система автоведения должна не просто обеспечить прибытие поезда на следующую станцию с учетом всех ограничений на скорость движения, условий сцепления колеса с рельсом, а сделать это при минимальном расходе электроэнергии или топлива. Ясно, что «примирить» сложность расчета и его быстроту нелегко. Достаточно сказать, что существующие программы, реализованные на больших ЭВМ, за одну минуту рассчитывают всего 5—10 км пути, что сравнимо со скоростью движения поезда и явно не подходит для системы автоведения. При решении этой задачи в МИИТе была разработана специальная методика расчета, основанная на применении аппарата теории оптимального управления, дающая нужный результат при использовании недорогого однокристального 16-разрядного микропроцессора с быстродействием 500 тыс. операций в секунду.

При управлении движением тяжеловесного грузового поезда значительные трудности создает необходимость учета

возможных перенапряжений, возникающих в межвагонных соединениях (автосцепках) при трогании, торможении и в моменты проезда изменений профиля пути.

Другая сложная задача — определение местоположения поезда. Традиционный способ заключается в подсчете импульсов, получаемых с осевого датчика,— их число пропорционально пройденному пути. Для коррекции ошибок измерения, вызванных, например, юзом или боксованием, используются так называемые точечные датчики, однако их установка и эксплуатация на магистральных железных дорогах — весьма дорогостоящее мероприятие. В последнее время разработаны и другие методы определения координат поезда, порой довольно неожиданные. Например, американские инженеры предложили измерять координаты поезда с помощью искусственных спутников Земли. Устанавливаемое на поезде приемное устройство, созданное на базе 32-разрядного микропроцессора, обрабатывает сигналы от двух спутников, находящихся на геостационарной орбите, и вычисляет местоположение. Испытания этой системы показали, что точность определения координат — около 45 м, что вполне достаточно для автоматизации многих функций, кроме, разумеется, торможения поезда на станции. Спутниковые каналы связи могут быть использованы и для передачи информации в централизованной системе автоведения.

Системы автоведения позволяют не только облегчить труд машинистов, но и улучшить работу железнодорожного транспорта. Испытания системы автоведения на Харьковском метрополитене показали, что только за счет более интенсивного торможения она значительно сокращает потери времени на остановку поезда. Этот резерв может быть использован для повышения интенсивности движения, для экономии электроэнергии на тягу. Внедрение систем автоведения повышает точность выполнения графика движения, что позволяет повысить использование пропускной способности линий за счет сокращения межпоездного интервала. И наконец, рационально выбирая режимы тяги, торможения, управляя поездами таким образом, чтобы не допускать их опасного сближения и как следствие этого дополнительных торможений, система автоведения может существенно уменьшить расход электроэнергии или топлива.

Уважаемые читатели!

Наш журнал еще очень молод и находится в стадии становления. Вы можете ускорить этот процесс, сделать журнал интересным и содержательным, ответив на наши вопросы.

Какие рубрики и материалы в ИНФО вам понравились, какие нет? Заинтересовала ли вас публикация «Мир ЭВМ» об операционной системе RT-11? Считаете ли вы, что журнал следует распространять не только по подписке, но и в розницу?

Какие новые рубрики вы хотели бы видеть? Например, заочную школу программиста; интервью с интересным человеком (с кем?); использование ЭВМ, предмет информатики, ее преподавание? Листок для родителей, который поможет им не отстать от детей, рефераты специальных статей по вычислительной технике?

Какие темы для научно-популярных статей вы хотите предложить?

Каково ваше мнение о курсе ОИВТ? Какие его части наиболее трудны для преподавания? Как его можно было бы усовершенствовать?

Ждем ваших писем!

Редакционная коллегия

Об использовании микрокалькуляторов при изучении курса «Основы информатики и вычислительной техники»

Во многих школах при изучении различных предметов (в том числе информатики) используются инженерные и программируемые микрокалькуляторы. Минпрос СССР получает большое количество писем, авторы которых спрашивают о методике работы с калькуляторами, проблемах снабжения ими школ, перспективах использования калькуляторов в учебном процессе.

Управление информатики и электронно-вычислительной техники Минпроса СССР, отвечая на эти вопросы, отмечает, что сейчас содержание курса «Основы информатики и вычислительной техники» составляет изучение алгоритмического языка, отражающего основные конструкции различных языков программирования. Умение записывать алгоритмы в виде удобных конструкций является наиболее трудной методической задачей курса — без этого умения нельзя сформировать активного пользователя современной техники. Язык микрокалькулятора по сравнению с алгоритмическим языком пробного учебного пособия менее нагляден, не поддается унификации, не позволяет продемонстрировать преимущества алгоритмического представления задачи. Это лишает калькулятор фундаментальной значимости в новом общеобразовательном предмете. Микрокалькулятор — эффективный заменитель логарифмической линейки и может быть с успехом использован в курсах математики, физики, других дисциплин.

Допускается использование микрокалькуляторов и на уроках информатики. Соответствующий методический материал включен в учебное пособие по информатике. Однако не следует рассматривать калькулятор в роли полноценной замены ЭВМ.

В перечень школьного оборудования пока включен только калькулятор МКШ-2. Остальные модели не удовлетворяют школьным требованиям, разработанным НИИ ШОТСО (наличие локальной сети, правила электробезопасности и т. д.).

Калькуляторы МКШ-2 заказываются школами, как и другое учебное оборудование, в установленном порядке.

В то же время Минпрос СССР предполагает не вводить централизованное снабжение школ различными калькуляторами для изучения курса информатики (по аналогии со счетными палочками, логарифмическими линейками).

Для тех школ, в которых вместо ЭВМ временно используются калькуляторы, методическое обеспечение занятий осуществляется минпросами союзных республик, сетью педагогических институтов, другими учреждениями. Управлением информатики и электронно-вычислительной техники разосланы методические материалы по работе с программируемыми микрокалькуляторами в союзные республики как возможную основу для методической проработки. По микрокалькуляторам предполагается опубликовать каталог методических материалов, изданных на местах.

Методические материалы по использованию микрокалькуляторов печатаются в журнале «Информатика и образование».

В министерства просвещения союзных республик разосланы брошюры «Проведение занятий по курсу «Основы информатики и вычислительной техники», подготовленные сотрудниками Куйбышевского авиационного института. Есть и другие издания, которые можно использовать в работе с микрокалькуляторами.

Журнал в журнале
для школьников, студентов,
учащихся СПТУ

Издается при участии
ЦК ВЛКСМ

МОЛОДЕЖНАЯ ИНИЦИАТИВА

1

В ЭТОМ НОМЕРЕ:

Проверь себя в трудном деле!

Наше время — время смелых решений, дерзких замыслов, конкретных дел.

Нам очень по душе предложение журнала «Информатика и образование» выпускать свою молодежную рубрику, которая со временем может дать новый импульс самодеятельности школьников, учащихся СПТУ, студентов в овладении новой техникой.

В ближайшие годы молодые специалисты самостоятельно создадут обучающие программы, помогут внедрить компьютеры в школе, профтехучилище, вузе, идейно закалятся и проверят свои силы в реальном деле.

Долгих тебе лет жизни, «Молодежная инициатива»!

Мы будет помогать твоим росткам пробиваться в будущее!

Секретарь ЦК ВЛКСМ *Л. И. Швецова*

Наши планы

Журнал в журнале для молодежи — инициатива молодых, отсюда и его название.

Мы будем разыскивать материалы, которые интересны вам. Подберем толковых ребят и создадим ученическую редколлегию или инициативную группу. Пока не знаем, как будет называться такая общность людей,— время покажет.

Выберем своего главного «инициатора». У нас самоуправление будет не на словах, а на деле. Только не надо забывать, для чего все это нужно. Компьютеризация — процесс неизбежный. Чем быстрее все мы освоим новые знания, тем быстрее и лучше пойдут дела у нас в селе, городе, стране.

Говорят что навыки и умения легче закрепляются в молодости. Будем учиться вместе. (Многим взрослым из ложного самолюбия стыдно задавать детские вопросы, а зря. Тогда пусть читают нашу «Школьную тетрадку».)

Конкурс

Я, КОМПЬЮТЕР, ПРОФТЕХУЧИЛИЩЕ

Госпрофобр СССР и Центральный совет ВОИР с 1 марта 1987 г. по 1 января 1989 г. проводят Всесоюзный конкурс на лучшее решение по созданию прикладных программ ЭВМ.
Учрежден 31 диплом с денежными премиями:
5 дипломов I степени с денежными премиями по 500 рублей;
8 дипломов II степени с денежными премиями по 300 рублей;
8 дипломов III степени с денежными премиями по 200 рублей;
10 поощрительных премий по 100 рублей.
В конкурсе могут участвовать специалисты и изобретатели независимо от ведомственной подчиненности, студенческие конструкторские бюро, кружки технического творчества, отдельные авторы.
На конкурс принимаются прикладные программы, тренажерные устройства с применением ЭВМ, педагогические программные средства для средних профессионально-технических училищ, педагогические сценарии программ и описания педагогических программных средств на уровне идей (методик обучения, проектов), пригодных для реализации на ЭВМ.
Образцы, представляемые на конкурс, должны отвечать требованиям техники безопасности и санитарной гигиены.
При разработке педагогических программных средств вычислительной техники необходимо ориентироваться на школьные персональные ЭВМ: «Электроника БК-0010» в комплекте с ДВК-2М (комплект учебной вычислительной техники «Электроника КУВТ-86»), «Корвет», КУВТ «Электроника МС 0202», «Ямаха», а также микро-ЭВМ «Электроника ДЗ-28».
На конкурс не принимаются разработки, созданные до 1986 г.
О требованиях к оформлению материалов.

Технические разработки. Представляются 2 экз. описания (экспонат не присылать) с экономическим обоснованием, а также тексты, схемы, чертежи. Необходима справка о внедрении или возможности использования в средних профтехучилищах и индустриально-педагогических техникумах.

Педагогические программные средства представляются на магнитных носителях. На конверте для диска должны быть указаны тип машины, язык программирования, операционная система, название программы, тип программы (демонстрационная, тренировочная, контролирующая, игровая и т. п.), потребность в памяти и периферийных устройствах, время прогона. Прилагаются инструкция пользователя, краткие методические рекомендации, указывается, где и кем сделана программа.
Материалы на конкурс направлять до 1 января 1989 г. по адресу:

125319, Москва, ул. Черняховского, д. 9. Всесоюзный научно-методический центр профессионально-технического обучения молодежи с пометкой «На конкурс ИВТ».

«Агат» напрокат

Неблизкий путь на Дмитровское шоссе, где расположился «Компьютерный центр» Лианозовского электромеханического завода, провожу в раздумье. Отличная идея, что и говорить,— дать всем желающим возможность узнать, что это такое — компьютер. Из короткого газетного сообщения известно, что, заплатив 1 руб. 20 коп. за час, любой человек — взрослый или ребенок — может самостоятельно позаниматься на «Агате» чем душе угодно: нарисовать график (не меньше чем для диссертации), составить каталог домашней библиотеки, наконец, просто поиграть в компьютерные игры. Только одолевают вопросы: а как здесь с

техникой безопасности; и выгодное ли это дело, не пустуют ли залы центра; и не заглохло ли вообще, не успев утвердиться, это начинание. Честно говоря, почему-то не верилось, что работают здесь строго по расписанию и что приеду и не окажусь перед закрытой дверью. Не приучены, что ли, к хорошему сервису? Да, видно так...

И перед основательной вывеской на доме № 115 по Дмитровскому шоссе, где сказано, что даже в субботу и воскресенье ждут нас в «Компьютерном центре», еще до конца не развеялись сомнения...

Но центр был открыт.

И помещения были просторны, и красиво оформлены были стены, и вежливы были сотрудники,

и у кассы, а также в зал компьютерных игр стояла небольшая очередь. А это, как известно, хороший признак. Сидели в зале программирования студенты, аспиранты, один рабочий из Александрова, специально приехавший в свой отпуск посмотреть и потрогать ЭВМ.

В зале для начинающих, где стоят «Микроши», было много школьников. Обнаружилось, что все они из пятых — седьмых классов, хотят самостоятельно научиться работать с компьютерами. Товарищи психологи, мотивация налицо!

Только жалко, что игры-то все с англоязычной «начинкой»! Их в центре 40 штук. (Справедливости ради, следует сказать, что в центре сознают серьезность этой проблемы и собираются предпринять шаги к исправлению положения.)

И вспомнился встреченный вчера в одной из московских школ увлекающийся программированием красивый десятиклассник Сережа, который составил игровую программу... «Бизнес» (?!). Не без гордости упомянул, что ввел условие: при землетрясениях и прочих чрезвычайных происшествиях курс... акций (!) меняется и т. д.

Не нравится мне ситуация с компьютерными играми, ребята!

Давайте вместе думать, школьники-программисты, студенты-программисты, учащиеся СПТУ-программисты. Похоже, пока сами не создадим хорошие программы (и обучающие, и игровые), дело не сдвинется с места!

А «Бизнес»... Биржа — это, конечно, любопытно, но не интереснее ли будет игра «Директор завода» — с учетом самофинансируемости и

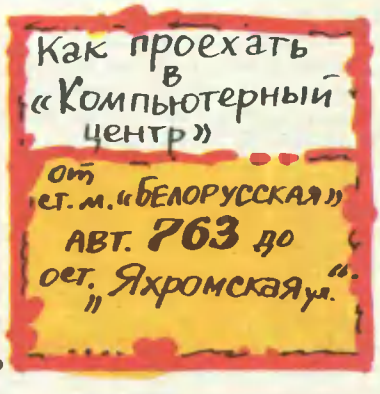
права внешнеторговых операций?

Но простите меня за это отступление. Итак, продолжим наше путешествие по «Компьютерному центру». Работает здесь в основном молодежь. Удобно то, что ремонт «Агатов» гарантируется предприятием. Хорошо бы, если бы велись статистические наблюдения, сколько отказов и по каким параметрам дают эти машины. И поскорее учитывать в работе завода полученные данные.

Весело светятся цветные дисплеи, приятно смотреть на оживленные лица ребят. Хорошее дело, приносящее пользу всем: государству, предприятию, нам с вами, — придумали на заводе!

А у вас в городе, селе, школе есть что-нибудь подобное?

Т. ДРАГНЫШ

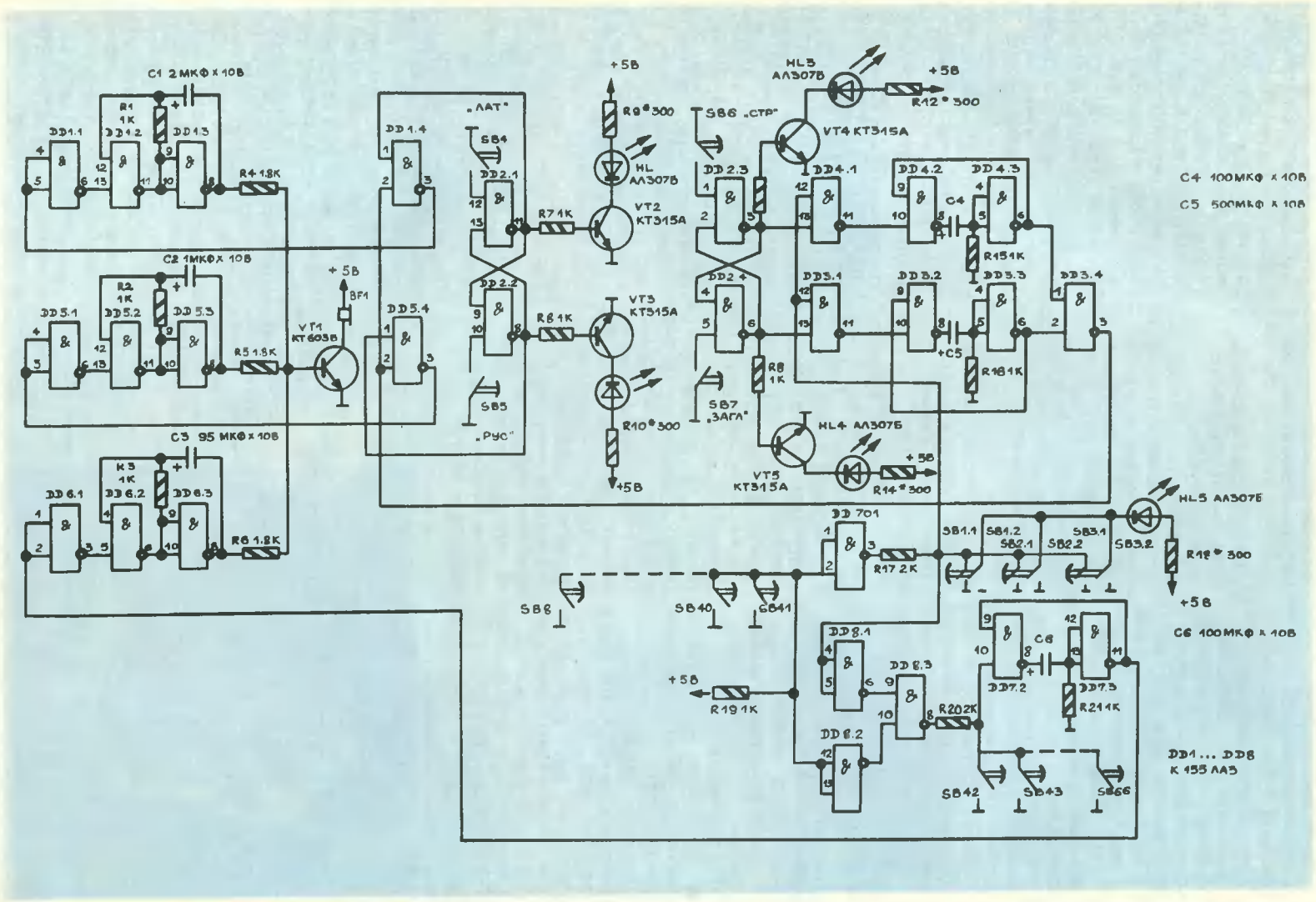


Центр работает ежедневно, кроме вторника, с 8.30 до 21.00 часа.
В воскресенье с 9.00 до 20.00 часов.
Его телефон: 485-31-34.

Изготовь тренажер сам!

Многие учебные заведения, особенно сельские школы, еще не имеют дисплейных классов, но приобрести начальные навыки работы на клавиатуре ЭВМ можно на тренажере, который разработан в нашей школе. Члены кружка электронного конструирования за короткое время изготовили и испытали первый образец. Изготовив 5—10 таких приборов, можно не только организовать работу с учащимися в классе, но и выдавать тренажеры на дом, особенно в классах с малой наполняемостью. Методика работы на тренажере чем-то напоминает метод обучения азбуке Морзе. На карточках составляются тесты разной сложности, в которых чередуются цифровые и алфавитные символы, строчные и заглавные русские и латинские буквы. Учащиеся работают парами. Один из них на клавиатуре тренажера набирает тест, второй его контролирует, подсчитывая допущенные ошибки. Затем они меняются местами. Учащиеся как бы соревнуются, кто быстрее и с наименьшим количеством ошибок наберет на клавиатуре данный текст. Позднее в качестве тестов можно использовать фрагменты готовых программ на том или ином языке программирования, а также «прогонять» на тренажере программы, составленные самими учащимися, перед тем как сесть за клавиатуру персонального компьютера. Играя, учащиеся изучают расположение символов на клавиатуре, приобретают навыки набора текста с высокой скоростью, привыкают пользоваться клавишами регистров.

Разрабатывая тренажер, мы стремились соблюсти следующие условия: он должен содержать распространенные недефицитные элементы общей стоимостью не более 10—15 руб.; изготовление прибора должно оказаться под силу начинающему радиолюбителю; конструкция клавиш должна быть очень простой и надежной, без применения заводских микропереключателей. Каким же образом, не имея перед собой дисплея, учащиеся догадываются о


C1 2МКФ х 10В
R1 1К
DD1.1
DD1.2
DD1.3
R4 1.8К
DD1.4
"АЛТ"
SB4
DD2.1
DD2.2
SB5
"РУС"
+5В
R9 300
HL АЛ307Б
VT2 КТ315А
R7 1К
R8 1К
VT3 КТ315А
R10 300
+5В
HL3 АЛ307Б
+5В
R12 300
SB6 "СТР"
VT4 КТ315А
DD2.3
DD4.1
DD4.2
DD4.3
C4
R15 1К
C2 1МКФ х 10В
R2 1К
DD5.1
DD5.2
DD5.3
R5 1.8К
+5В
VT1 КТ603Б
BP1
DD5.4
DD3.1
DD3.2
DD3.3
DD3.4
+C5
R16 1К
R8 1К
SB7 "ЗАГА"
HL4 АЛ307Б
VT5 КТ315А
+5В
R14 300
C4 100МКФ х 10В
C5 500МКФ х 10В
C3 95 МКФ х 10В
R3 1К
DD6.1
DD6.2
DD6.3
R6 1.8К
DD 701
SB1.1
SB1.2
SB2.1
SB2.2
SB3.1
SB3.2
HL5 АЛ307Б
R18 300
+5В
R17 2К
SB8
SB40
SB41
DD8.1
DD8.3
DD8.2
+5В
R19 1К
R20 2К
C6
DD7.2
DD7.3
R21 1К
C6 100МКФ х 10В
DD1 ... DD8
К 155 АА3
SB42
SB43
SB44

том, правильно ли с клавиатуры введены символы? Для этого используется двойная индикация — звуковая и световая. Четыре светодиода индицируют состояние клавиш *стр*, *загл*, *рус*, *лат*, если предварительно, пусть даже кратковременно, нажать любую из них, то загорается соответствующий светодиод и учащиеся видят, какие символы вводятся с клавиатуры.

Кроме того, каждому типу символа соответствует свой звуковой сигнал (в динамике или головных телефонах, подключаемых параллельно динамику). Их частоты и длительность некритичны. Главное, чтобы сигналы, соответствующие разным видам символов, хорошо различались на слух. В нашем устройстве они таковы.

Для заглавных русских букв — импульсы частотой 400 Гц большой длительности.

Для строчных русских букв — частотой 400 Гц, а длительность меньше примерно в три раза. Здесь как бы проводится аналогия: заглавная буква больше размером — и сигнал звучит больший промежуток времени. Строчная буква меньше размером — звучит короткий сигнал той же частоты.

Для латинских заглавных букв частота звукового сигнала 1200 Гц, длительность такая же, как для заглавных русских букв. Для латинских строчных — короткий сигнал той же частоты (1200 Гц).

Для звуковой индикации цифровых символов служат короткие сигналы частотой 3600 Гц.

Для того чтобы с клавиатуры БК-0010 ввести символы правого или нижнего регистров, служат специальные клавиши *пр* и *нр*, для «ввода» соответствующих символов одной рукой нужно удерживать эти клавиши в нажатом состоянии (о чем говорит загоревшийся специальный светодиод под надписями и на передней панели), а другой рукой нажать клавишу, на которой изображен нужный символ (при этом появится короткий сигнал частотой 3600 Гц). В схеме задействованы в основном цифровые, алфавитные клавиши, клавиши *пробел* и *ввод*, т. е. те, которые нужны для набора программы. Клавиши курсора и другие вспомогательные не распаяны.

Принципиальная электрическая схема приведена на рис. 1. Мы не стали подробно описывать логику работы микросхем. Об этом можно прочитать в журнале «Радио» за 1985 г. в номерах 1, 2, 3, 5, 7, 8, 9, авторы В. Борисов, А. Партин, статья «Основы цифровой техники», раздел «Практикум начинающих»

На элементах DD 1.2 и DD 1.3, DD 5.2 и DD 5.3, DD 6.2 и DD 6.3 собраны мультивибраторы на частоты 400, 1200, 3600 Гц соответственно.

На элементах DD 4.2 и DD 4.3, DD 3.2 и DD 3.3, DD 7.2 и DD 7.3 собраны ждущие мультивибраторы, которые формируют импульсы определенной деятельности для запуска того или иного мультивибратора, генерирующего звуковые сигналы указанных выше частот. Запоминание нажатия регистровых клавиш *лат* и *рус* осуществляется с помощью простейшей ячейки памяти — тригера на элементах DD 2.1 и DD 2.2. Светодиоды HL1 и HL2 индицируют состояние этих клавиш.

По аналогичной схеме (на элементах DD 2.3 и DD 2.4) собрана ячейка памяти для запоминания и индикации нажатия регистровых клавиш *стр* и *загл*. Все остальные элементы микросхем служат в основном для инвертирования сигнала в соответствии с логикой работы устройства.

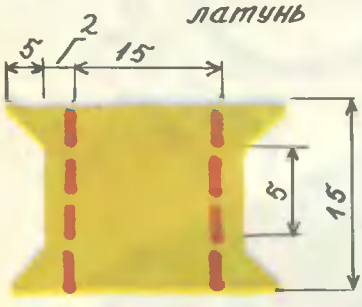
Кнопки SB 8... SB 41 — алфавитные клавиши; кнопки SB 42... SB 66 — цифровые и некоторые специальные клавиши, а также четыре клавиши пробела.

Кнопки SB1, SB2, и SB3 — клавиши *пр*, *нр* и *су*. Эти кнопки выполнены

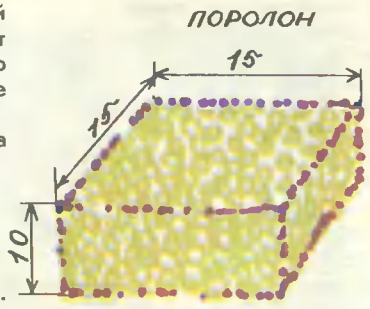
Анатолий Иванович работает учителем в средней школе поселка Речной Куменского района Кировской области. Он и его кружковцы покупают недорогие микросхемы через Посылторг и мастерят интересные самоделки

сдвоенными. При удерживании в нажатом состоянии любой из них один из замкнутых контактов разрешает работу мультивибратора частотой 3600 Гц, а другой замыкает цепь индикации нажатого состояния на светодиоде HL 5.

Верхняя панель тренажера выполнена из листа дюралюминия толщиной 1—1,5 мм. Эта металлическая панель соединяется с общим проводом (минус источника). Самодельные клавиши, установленные на этой панели, по размерам, цвету и расположению напоминают ту клавиатуру, на которой будут работать учащиеся (в данном случае — клавиатуру БК-0010). Конструкция клавиш проста. Из латунной полоски толщиной 0,5 мм вырезается заготовка в соответствии с чертежом (рис. 2). Затем концы



заготовки загибаются под прямым углом по пунктирным линиям. Это и будет подвижный контакт клавиши. Применяя приспособление с разогреваемой нихромовой проволокой, нарезают кусочки поролона по размерам в соответствии с рис. 3. Размечают панель прибора, приклеивают клеем «Момент» заготовки из поролона, а на них тем же клеем — подвижные контакты. Предварительно к одной из четырех «ножек» каждого контакта нужно



припаять тонкий гибкий многожильный провод, который затем проходит сквозь панель через отверстие ∅ 3 мм. После сборки клавиши на подвижные контакты наклеивают соответствующие символы, которые желательно покрыть прозрачной пленкой. Для изготовления сдвоенной клавиши подвижный контакт распиливают пополам, а затем наклеивают на тонкую гетинаксную пластинку (с зазором между половинками 1 мм).

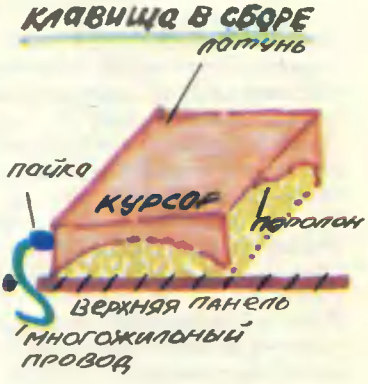
Элементы C 4, R 15, C 5, R 14, C 6, R 21 определяют длительность звуковых сигналов.

Емкость конденсаторов C1, C 2, C 3 определяет частоту мультивибраторов.

Световоды HL 1, HL 2, HL 3, HL 4 включены через транзисторные ключи, чтобы при отсутствии светодиодов на их место можно было ставить миниатюрные лампы накаливания.

При наличии светодиодов все их можно включать и без транзисторных ключей (подобно светодиоду HL 5). В зависимости от применяемых светодиодов следует подобрать номиналы регистров R 9, R 10, R 12, R 14, R 18. Вместо КТ603Б (VT1) можно использовать КТ 815, КТ 801 с любым буквенным индексом.

На передней стенке устанавливается разъем для подключения стандартного блока питания (от микрокалькуляторов «Электроника Б3-34, Б3-18 А» и др.).



В качестве BF1 можно использовать малогабаритную динамическую головку от старого транзисторного приемника или головные телефоны.

КАРАВАЕВ А.
Учитель физики и информатики, руководитель кружка электронного конструирования, заслуженный учитель школы РСФСР

Паскаль для избранных

Ну вот, кажется, пришел конец спорам о том, какой язык программирования следует изучать в школах, — издан русский перевод* книги Р. Форсайта «Pascal at work and play». Раз уж Паскаль пригоден и для игры, и для работы, то и для учебы он — лучше не придумаешь!

А если серьезно?

«То, что Паскаль дает возможность справиться со множеством разнообразных задач, является одним из самых сильных аргументов в пользу его применения», — говорит автор; а то, что Паскаль позволяет справиться, наглядно показано на страницах книги.

Одна из ее характерных черт — использование в качестве примеров (достаточно многочисленных) не игрушечных, как это обычно делается, а вполне серьезных, разрастающихся к концу книги до сотен операторов программ, решающих практически полезные задачи. Программа поиска оптимального маршрута в лондонском метро не менее запутанна, чем сам метрополитен (схема которого приведена на форзаце), — но это впечатление, возникающее при первом просмотре, исчезнет, если ревностный читатель внимательно прочтет предшествующий текст и выполнит предложенные упражнения; а программа для игры в футбол покажется ему не менее увлекательной, чем сама игра.

Однако, несмотря на сложность, это лишь упражнения. «Автор считает, что целью является получение результатов, помогающих решить стоящую перед ним задачу. Программа является просто средством, позволяющим добиться этой цели, а построение алгоритма — это одна из стадий, обязательная при составлении программы».

Книга не свободна от опечаток, по-видимому, неизбежных в такого рода изданиях, однако они не портят общего впечатления, чего нельзя сказать о русском варианте названия. «Паскаль для всех» — тиражом 50 тысяч экземпляров?! Спрос на подобную литературу явно недооценивается. Кстати, говорят, на днях продавалось новое руководство по Паскалю — издательства «Мир». Не надите почитать?

ШЕХОВЦЕВ К.

* *Форсайт Р.* Паскаль для всех. М.: Машиностроение, 1986.

Юные программисты: первые успехи

В Переславле-Залесском успешно прошла зимняя школа юных программистов. Среди отличившихся школьников — девятиклассник школы № 23 г. Горького Валерий Фокин, сумевший лучше других построить программу, моделирующую поведение группы роботов в сложных условиях окружающей среды. Диплома первой степени удостоен и переславский девятиклассник из школы № 6 Андрей Кук. В группе младших отмечены работы шестиклассника этой же школы Саши Кука и четвероклассника Тимура Первина (школа № 6, г. Переславль), москвички-второклассницы Вики Масловой. Этим ребятам наиболее удались машинные рисунки, запрограммированные на «Лого» в виде серий вложенных процедур. Их друзья по группе с увлечением манипулировали световым пером на цветном экране дисплея, работали с программными моделями простых роботов-исполнителей. Шестиклассник из Москвы Аркадий Терешенков, освоивший «Лого» в дни работы зимнего лагеря, предложил алгоритм, использующий разнообразные средства программной системы — от динамической графики до датчика случайных чисел, и реализовал этот алгоритм в ходе учебных занятий.

Э. Н. ЕРМАКОВ

Школьное сочинение на тему, подсказанную читателем

В № 1 за 1987 г. журнал «Информатика и образование» поместил фотографию десятиклассника, составившего интересную игровую программу. И сразу же мы получили предложение молодого челябинского учителя информатики С. М. Пястолева заказать автору программы «Ним» статью, поскольку школьники из Челябинска работают над подобными программами и им было бы интересно обменяться опытом со своим сверстником. Выполняем просьбу читателей.

Я учусь в X классе школы № 625 в Москве. Одной из причин, по которой я пошел в эту школу, была информатика, которая преподается в нашей школе с самого ее открытия. К тому же два года назад у нас оборудовали компьютерный класс, оснащенный ПЭВМ «Агат». Несмотря на то что

эта машина имеет недостатки (частые сбои в оперативной памяти, очень ненадежный дисковод, часто ломающуюся клавиатуру), на ней можно хорошо постичь азы программирования. Мы изучаем реальные языки: Бейсик и Рапиру. Рапира по структуре имеет большое сходство с Паскалем. Пока «Агат» не может похвастаться большим программным обеспечением. Поэтому ребятам, которые хорошо освоились в программировании, наши преподаватели (сотрудники ВЦ АН СССР А. О. Гончаров и М. И. Селюн) дают крупные работы, рассчитанные на несколько месяцев. Это и игровые программы, и работы, предназначенные для специального обеспечения пакетами прикладных программ (по математике, физике и даже по истории, обществоведению) и программами технического обслуживания. Например, Илья Борисов сделал в прошлом году программу «Музыкальный редактор», которая предоставляет возможность написать или быстро воспроизвести различные мелодии. До этого на «Агате» было очень сложно написать музыку.

Вообще в IX классе, когда мы работали на Бейсике, была сделана не одна интересная работа: Уже упомянутый «Музыкальный редактор» Борисова, первый текстовый редактор на «Агате», разработанный Виктором Глуздовским, игровые программы «Pexeso» (автор Сергей Чугунов) и моя игровая программа «Ним», представляют собой математическую головоломку. Также ребята программировали методы решений уравнений и их систем: методы Гаусса и итерации для решения систем линейных уравнений, методы деления отрезка пополам и хорд для решения нелинейных уравнений и др.

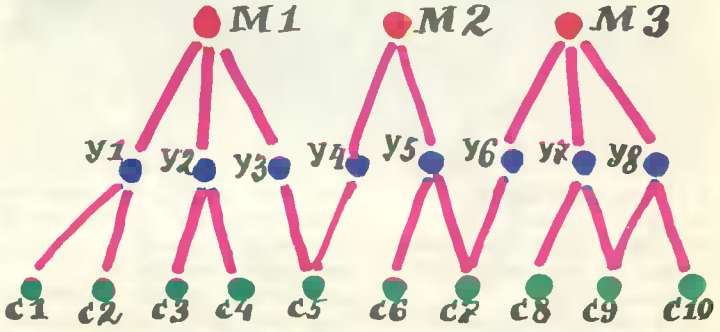
В X классе количество готовых больших работ уменьшилось, так как большинство ребят повторяют к экзаменам пройденные темы и составляют по ним короткие программы. Но есть и большие разработки.

Например, Сергей Наумов сделал большую игру «Лабиринт», я недавно закончил «Журнал», дающий возможность быстро получить все необходимые сведения об ученике, помогающий объективно выставить оценки.

В электронном журнале фамилии учащихся можно расставлять не только по алфавиту, но и по возрастанию (или убыванию) среднего балла, по количеству оценок. Это помогает установить очередность, в которой следует опрашивать учеников. При необходимости учитель сможет вывести на дисплей все сведения об успеваемости ученика: даты опросов, темы занятий в этот день или формулировки индивидуальных заданий, вопросов.

Одновременно с уроками информатики в школе вот уже в течение года я работаю на ВЦ АН СССР под руководством нашего преподавателя О. А. Гончарова. За это время я составил пакет, состоящий из восьми программ «Персоплан» (персональный планировщик), который предназначен для гибкого сетевого планирования как в индивидуальной деятельности человека, так и для помощи в планировании деятельности коллектива.

Теория такого планирования составлена А. Г. Шмелевым. В одном из частных случаев сетка представляет собой такой вид:



где У — цели, которые преследует человек или коллектив. С — средства, необходимые для достижения целей, и М — мотивы или более высокие цели, интересы. План состоит из путей, располагающихся в определенном порядке. Путей столько, сколько средств. При подсчете коэффициентов к ним учитывается трудность выполнения средства, срочность целей, важность мотивов и количество целей и мотивов, достигающихся после выполнения средства. Для существенного увеличения гибкости я разработал корректировочную программу, которая может удалить из плана выполненные средства (цели или мотивы).

ЭПШТЕЙН Л.,
X Б, школа № 625, Москва

Комсомол и компьютерный всеобуч

Участники СНПО на ВДНХ СССР.

Введение основ информатики и вычислительной техники в школьное обучение требует создания соответствующего программного обеспечения — прикладных обучающих программ. Объемы и сложность предстоящей работы громадны. Кто же поможет школе, педагогам быстро и качественно решить важную задачу? Нельзя ли применить хорошо зарекомендовавшие себя формы организации творческой деятельности студенческой и учащейся молодежи и на этот раз? Поиску ответов на эти и другие важные вопросы компьютеризации школы был посвящен проходивший в феврале нынешнего года в Москве семинар представителей вузов страны. Если быть более точным, темой семинара-совещания стал разговор о том, как организовать помощь комсомольцев-студентов и учащихся педагогам, психологам, социологам, другим специалистам, работающим над созданием педагогических программных продуктов для ЭВМ в школах и профессионально-технических училищах.

Занимаясь подготовкой и проведением самого семинара и позднее — начиная эту статью, мы не раз ловили себя на мысли о том, как не хватает нам уже сегодня хорошего, работящего и безотказного помощника — компьютера. Думали о нем и тогда, когда, обложившись справками, газетными и журнальными вырезками, статьями об участии комсомольских организаций в привлечении учащихся к овладению основами информатики и вычислительной техники, проведении Всесоюзной школы молодых ученых и специалистов в Тбилиси, семинара «Народные университеты и

компьютерное образование» в Воронеже, материалам «круглого стола» по проблемам компьютеризации в НИЦ ВКШ ЦК ВЛКСМ, старались подготовить для участников семинара материал, который бы наиболее полно отразил деятельность комсомольских организаций по обеспечению компьютерного всеобуча и, вызывая из своей памяти, но не из памяти ЭВМ детали прошедшего, еще раз воспроизводили строчки, уже написанные однажды нами самими в том или ином документе.

Думали и тогда, когда пришлось не один десяток раз звонить по телефону для уточнения состава участников, даты, времени их прибытия в Москву, беспокоились об обратных билетах и множестве других организационных мелочей.

Не скроем — приходила на ум и фантастическая, а может и реальная, мысль о единой информационной сети, которая в одночасье разгрузила бы нас всех от рутинной, механической работы и позволила бы более предметно заниматься практическими творческими делами. Действительно, располагай мы такой сетью, скажем, между вузами,— уже к концу первого дня работы Московского семинара СНПО все вузовские комитеты комсомола (а не 57 из них) знали бы не только о том, что на семинаре выступили начальник Управления информатики и электронно-вычислительной техники Минпроса СССР А. Ю. Уваров, его коллега из Госкомитета по профтехобразованию В. О. Хорошилов, ответственный секретарь научно-технического совета по НИРС Минвуза СССР В. Л. Лагерев, заместитель генерального директора хозрасчетного объединения Минвуза РСФСР И. В. Иванов и многие другие товарищи, но и, получив тезисы выступлений участников семинара, смогли бы включиться в общий разговор по вопросам организации деятельности творческих молодежных объединений. Тем более что мнения участников семинара на этот счет не были абсолютно одинаковыми.

Согласившись, к примеру, в принципе с идеей участия студентов в разработке программного продукта, начальник Управления информатики и электронно-вычислительной техники Минпроса СССР А. Ю. Уваров высказал сомнение в способности студенческих отрядов выполнить эту задачу без совместной работы со школьными учителями. Такое ви́дение проблемы вызвало оживленную дискуссию, во время которой с достаточной степенью доказательности, приведением конкретных примеров в пользу работы студентов и учащихся над созданием ППС высказались представители московских институтов электронного машиностроения, физико-технического, Свердловского пединститута и большинства других вузов. Правда, в ходе обсуждения совершенно однозначно было признано, что такая деятельность студентов должна организовываться с привлечением профессорско-преподавательского состава и обязательно в тесном взаимодействии с опытными педагогами-предметниками и психологами.

Принятые по итогам семинара рекомендации организаторам СНПО как раз и советуют активно привлекать их к работе в отрядах.

В Министерстве просвещения СССР подготовлено письмо в адрес республиканских, краевых, областных отделов народного образования, рекомендующее им обеспечить такое сотрудничество педагогов-практиков, ученых, специалистов и участников студенческих отрядов. В итоговом документе участники семинара-совещания обратились с просьбой в адрес Министерства просвещения СССР, Госкомитета по профтехобразованию, Академии педагогических наук СССР, соответствующих органов на местах сформулировать основные требования и тематику педагогических программных средств, необходимых общеобразовательной и профессиональной школе, выступить в этом случае в роли заказчиков для СНПО. По инициативе участников во время семинара при Центральном штабе студенческих отрядов ЦК ВЛКСМ создан Временный координационный совет по организации СНПО

разработчиков педагогических программных средств для школы и СПТУ. Координационный совет призван изучить, обобщить и распространить опыт творческих молодежных коллективов по созданию ППС, обмен программными средствами, методическими материалами. Совет занимается изучением проблемы материально-технического обеспечения отрядов, поиском эффективных механизмов морального и материального стимулирования деятельности участников отрядов, организацией научно-методической помощи со стороны ученых и специалистов. Поддержку получило и предложение о проведении в 1988 г. Всесоюзной школы молодых ученых и специалистов по проблемам создания педагогических программных продуктов для школьных ЭВМ.

Одна из важных тем, обсуждавшихся на семинаре,— в каких вузах предпочтительнее создавать отряды по разработке программного обеспечения — в технических или педагогических и университетах, какими быть студенческим отрядам по составу.

Однозначного ответа на эти вопросы дать, видимо, нельзя, однако и это нашло отражение в рекомендациях семинара,— преимущество в организации СНПО разработчиков педагогического программного продукта остается все-таки за университетами и педагогическими вузами. Ведь здесь студенты помимо навыков программирования получают знания по педагогике, общей и возрастной психологии, общаются с детьми. Именно в эти вузы в первую очередь приходят люди, чувствующие призвание учить, призвание воспитывать. Не менее важен и еще один аспект — собственно подготовка учителя, учителя современной школы. Специалисты отмечают, что правильная постановка НИРС — это одно из наиболее эффективных средств воспитания будущих учителей, способных творчески применять в практической деятельности последние достижения научно-технической революции.

В один из дней работы семинара его участники посетили ВДНХ СССР, где на Московской областной выставке НТТМ познакомились с опытом работы научно-производственного отряда МФТИ «Спектр». Рассказывая об отряде, его командир — аспирант Московского физико-технического института В. Фомин, и директор выставки Е. Островский выделили два этапа в деятельности творческого объединения. На первом этапе, с появлением в институте японских «Ямах» из желающих (в основном это были первокурсники) был сформирован коллектив, который в течение семестра создал около 30 программ. Среди первых программ — игровые («ксоникс», «рендзю»), обучающие («английский язык», «аккумулятор», «гравитационное поле»), а также аппаратные средства («редактор спрайтов», «доступ к видеопамяти», «пересылка двойных файлов»). Были выработаны задачи, поставленные затем перед СНПО: создать комплекс обучающих программ для средней школы и вуза, интеллектуальные средства для разработки прикладных программ, подготовить для работы с компьютерами возможно большее число студентов МФТИ и отладить организационные формы деятельности отрядов разработчиков ППС, вести научно-техническую пропаганду, поднять научный престиж работы среди студентов института, найти новые формы выявления наиболее способных школьников и привлечения их в вуз. Широкий спектр задач, решаемых отрядом, и обусловил его название.

С лета прошлого года «Спектр» работает на первом отечественном бытовом компьютере «БК-0010». Это связано с тем, что на «Ямахах» работает несколько СНПО, а вот программный продукт для «БК-0010» очень беден и нуждается в скорейшем пополнении.

«Спектр» заключил договор о научно-техническом сотрудничестве с заводом-изготовителем компьютеров «Электроника БК-0010».

В июне на физтех прибыл класс из 12 «Электроник» и ДВК, а к началу октября партия из 10 игровых

и учебных программ, написанных отрядом, была передана на завод. В это же время отряд успешно сотрудничал с Московским областным институтом усовершенствования учителей в написании учебных программ для школы. Последнее время «Спектр» выполнял заказ Московского историко-архивного института по автоматизации проводимых у них деловых игр, помогал в размещении техники в школах Московской области, члены отряда преподавали программирование своим младшим товарищам. У читателя может сложиться впечатление, что в деятельности отряда нет проблем. Это не так. Первое и самое главное. не решен вопрос с оплатой. Очевидно, что необходимы хоздоговорные лимиты. Вторая серьезная проблема — нехватка машин. Третья трудность — низкая надежность техники. Но, самое главное, есть желание работать, давать практические результаты.

«Изучая опыт работы первых СНПО, можно сделать вывод о высокой эффективности и полезности таких формирований. Они наилучшим образом позволяют сочетать учебную, научно-производственную и общественно-политическую деятельность студентов» — такими словами завершил выступление на ВДНХ командир «Спектра» Владимир Фомин.

«На базе НИИЯФ МГУ группа профессора А. Т. Рахимова создала специальный компьютер, который был рекомендован Минвузом и Минпросом СССР для установки в двенадцатой пятилетке во многих школах нашей страны. Фактически перед отрядом стоит задача материального обеспечения учебно-производственного комбината Ленинского района. В перспективе — установка этих машин непосредственно в школах Ленинского района» — эти строчки из многотиражной газеты «Московский университет» могли бы послужить своеобразным эпиграфом следующей темы разговора на семинаре — участие студентов и учащихся в разработке и изготовлении электронно-вычислительной техники, ориентированной на использование в школах и СПТУ.

Сегодня этим увлекательным и полезным делом заняты студенты Москвы и Киева, Харькова и Свердловска, многих других городов страны. В большинстве случаев работа творческих групп организуется по принципу студенческих отрядов. Это направление в деятельности студентов привело к еще одному самостоятельному и набирающему размах, а поэтому требующему особо пристального внимания Министерства просвещения СССР, Академии педагогических наук, других компетентных организаций явлению — созданию передвижных обучающих электронно-вычислительных классов и комплексов. Они активно действуют в Свердловске и Горьком, Гомеле и Харькове, Ереване и на Ставрополье. Действуют, несмотря на громадное число проблем, неопределенность статуса, причем количество их год от года увеличивается. В качестве передвижных средств для своих классов ЭВТ участники студенческих отрядов и молодежь использовали уже практически все, что можно было использовать,— автобусы, поезда, пароходы, вертолеты и самолет. Техника применяется при этом самая разная, преимущественно отечественного производства, и работает даже в таких сложных условиях. На нынешнем этапе, видимо, нецелесообразно и дальше делать вид, что явление это частное, а надо последовать примеру Минпроса РСФСР: серьезно заняться изучением этой работы, поддержкой молодежной инициативы, распространением лучшего опыта. А опыт такой уже есть. Могли бы оказать помощь и молодежные творческие коллективы, к примеру, такие, как ВМТК «Компьютерный центр» Свердловского государственного ордена «Знак Почета» педагогического института, отряд «Компьютерный десант» Горьковского обкома комсомола и другие.

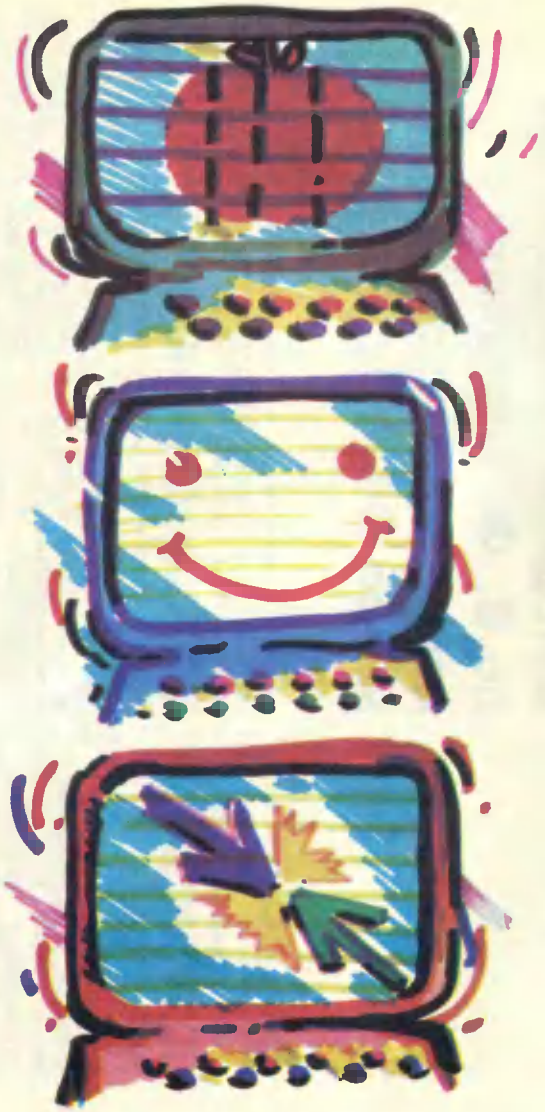
ПАРАМОНОВ А., ЛИСОВ В.

1.
2.
3.

Составление программы не должно быть самоцелью, поскольку программа — средство, а не цель обучения.

Позитивное подкрепление стимулирует освоение учебного материала.

Моделирование реальной ситуации предпочтительней манипулирования абстрактными понятиями.



Советы психолога

ОТ ДИПЛОМНОЙ РАБОТЫ — К БУДУЩЕЙ КНИГЕ

Программная поддержка методики преподавания по В. Ф. Шаталову для класса персональных микро-ЭВМ

Одним из наиболее эффективных современных способов решения проблемы интенсификации обучения является методика известного донецкого педагога В. Ф. Шаталова. Ядром этой системы является представление теоретической информации, подлежащей усвоению, в виде тщательно подобранных опорных сигналов — структурированных графических образов. Представляет интерес попытка соединения достоинств методики обучения по В. Ф. Шаталову с педагогическими возможностями класса персональных микро-ЭВМ. Описываются функции разработанного пакета прикладных программ. В моей дипломной работе рассматривается методика проведения занятий в компьютерном классе с использованием данного педагогического программного средства. Анализируются достоинства и недостатки обучения по этой методике в сравнении с постановкой учебного процесса по В. Ф. Шаталову и традиционной методикой.

ЩЕГОЛЕВ А.,
выпускник МИЭМа

А. Щеголев сам сформулировал основную идею своей дипломной работы. Остается добавить, что он на общественных началах является заместителем директора компьютерной физико-математической школы при МИЭМе, в которой учатся 600 старшеклассников. С марта этого года он участвует в работе студенческого научно-производственного отряда «Информатика», который будет разрабатывать программы для школьных компьютеров. Ждем информации о конкретных результатах работы отряда.

Придумала и отредактировала журнал в журнале — *Т. Драгныш*

Макет и оформление — *А. Архутика, А. Лобашинского.*

Иллюстрации к репортажу из «Компьютерного центра» — *Э. Бажилина.*

ПЕШЕЛЬ М.
профессор
Центр научного приборостроения АН ГДР

Для чего нужна информатика в школе

В настоящее время никто уже не сомневается в том, что овладение вычислительной техникой и ее широкое использование стали определяющими факторами научно-технического прогресса во всем мире.

В 90-е годы большинство профессий будет прямо или косвенно нуждаться в использовании вычислительной техники, для того чтобы рационально использовать в соответствии с общественными запросами важнейший человеческий ресурс — рабочее время.

Наряду с материальными и энергетическими сейчас возникает информационный ресурс вместе с традиционными фундаментальными дисциплинами — математикой, физикой, химией, биологией в эпоху микроэлектроники, информатика в сочетании с кибернетикой, системным анализом приобретает большое значение в деле революционного преобразования жизни и способа производства общества.

При этом значение информатики при ее проникновении в большинство профессий выходит за рамки роли классических дисциплин, так как для многих профессий возникает необходимость использования информационно-технических средств.

Мы должны сообщить школьникам и студентам новые знания и навыки, возникающие в результате стремительного развития микроэлектроники, вычислительной техники, если хотим в 90-е годы использовать способности молодых как важный образовательный и производственный фактор.

Изучение информатики в школе рассматривается во всех высокоразвитых странах как ключевой вопрос, и каждая страна старается, с учетом своих особенностей, удовлетворить требования времени к школьному образованию. Понятно, что процесс интеграции будет происходить так, что изменится облик всех учебных предметов. Однако путь, по которому должен идти этот процесс, остается еще неясным и неопределенным. Эта статья направлена на поиски решения вопроса о месте информатики в школе.

Сначала сформулируем нашу точку зрения на то, что означает понятие «Информатика в школе».

Под информатикой в школе мы понимаем ее включение в школьное образование. Причем данное положение особенно относится к общеобразовательной школе. При этом включение не означает введение дополнительного предмета «Информатика», а внутреннюю связь материала каждого предмета с возможностями вычислительной техники, рационализация и интенсификация процесса сообщения знаний без дополнительной временной нагрузки на учителей или школьников.

Информатика — междисциплинарная научная область, на которую должны ориентироваться все дисциплины с точки

зрения их общественного использования. Так должно быть в любом предмете с учетом его специфики. Было бы плохо, если бы прочие предметы преподавались как и прежде, а вместе с ними был введен дополнительный предмет «Информатика». Процесс установления взаимосвязей и объединения знаний был бы переложен на самих учащихся.

Если нам удастся широко использовать вычислительную технику, то мы сможем значительно ускорить научно-технический прогресс, повысив уровень выпускаемой продукции. Можно будет автоматизировать рутинные операции, что позволит получить больше времени и сил для творческой работы.

Применение ЭВМ ведет к повышению качества работы. Информатике предвещают положение одного из главных культурных навыков человека — после чтения, письма и счета, а компьютеризацию обучения называют второй грамотностью человечества.

Сейчас в СССР широко внедряется электронная вычислительная техника в учебный процесс. Создан новый журнал «Информатика и образование», и активно сотрудничать с ним могут ученые и педагоги социалистических стран.

Академик А. П. Ершов, руководитель отдела информатики в Вычислительном центре Сибирского отделения АН СССР, обрисовал в одном из интервью, каким образом в ближайшие годы в СССР будет происходить процесс введения компьютеров в школе. Перед нами стоит задача проведения в ГДР, в соответствии с экономическими возможностями, аналогичного процесса.

Для этого Политбюро СЕПГ в 1985 г. приняло важное решение о развитии информатики в образовании, а XI съезд СЕПГ дал необходимые ориентиры этой работы.

Надо отметить, что благодаря широкой инициативе нашего народного образования в союзе с Академией педагогических наук ГДР мы уже достигли многообещающего уровня по созданию принципов компьютерного обучения и материально-технической базы в народном образовании. С 1985/86 учебного года электронный калькулятор введен в ГДР как обязательное техническое средство для работы учащихся начиная с VII класса. Такой калькулятор позволяет вводить элементы алгоритмического мышления в процесс обучения, в частности на уроках математики. С его помощью можно рационализировать процесс счета, добиваясь алгоритмического мышления учащихся как цели обучения, относящейся ко многим предметам.

Существенно повысить эффективность компьютерного обучения могут персональные компьютеры (ПК). Они должны быть включены начиная с определенного класса в учебный процесс как средства рационализации в самых различных предметах. ПК даже в первых своих формах исполнения представляет широкий спектр возможностей и в том, что касается графических функций. Он гибок в управлении, и при его включении в учебный процесс можно говорить о значительных качественных изменениях в системе образования, которые затронут все предметы.

Цели обучения в общеобразовательной школе и информатика

Математика, физика, химия и биология, общественные науки являются теми традиционными предметами, благодаря изучению которых каждому учащемуся сообщаются определенные фундаментальные знания в рамках общего образования. В этом процессе наша система образования достигла значительных успехов, и ее опыт распространился широко за пределами ГДР.

В эпоху развития и роста возможностей микроэлектроники наряду с традиционными основными дисциплинами в школьном образовании выступают новые междисциплинарные предметы, такие, как кибернетика и информатика. Поэтому надо тщательно проверить, в какой мере общеобразовательная школа, формируя новые и поддерживая прежние цели образования, соответствует новым требованиям.

Достаточно ли общее школьное образование ориентировано на главные направления научно-технического прогресса?

Отчетливо ли учитываются цели образования — сделать школьников способными и мотивировать их для самостоятельного продолжения образования в течение всей жизни?

Как концепция образования учитывает те методы мышления, работы, методические принципы, которые присущи научным дисциплинам, лежащим в основе соответствующего предмета?

Введение информатики в общеобразовательную школу связано с формированием новых целей образования и будет способствовать их достижению. Ниже содержатся частичные ответы на этот важный вопрос.

Процесс обучения по большинству предметов может получить конкретную практическую форму, так как запасы знаний могут быть использованы в более гибких формах, чем это сделает учитель с помощью учебников.

Взаимные связи между различными предметами возможны за счет объединяющего характера информатики, более отчетливо в смысле единства наук. Благодаря графическим возможностям персонального компьютера возникают новые формы представления знаний, в дополнение к школьной доске и диапроектору ПК позволяет проводить по многим естественнонаучным предметам эксперименты методом моделирования. Передача заданий становится более динамичной, что ведет к переосмыслению и новому формированию материала с необходимыми выводами для разработки новых учебников.

Игровой фактор становится благодаря экспериментам с ЭВМ важной компонентой передачи знаний, что повышает привлекательность предметов.

Правдивость как важная компонента формирования личности является обязательной предпосылкой эффективной работы с компьютером. Хотя компьютер и делает арифметические ошибки, однако в отношении своей «логики» он обычно всегда прав.

Последствия информатики

В ближайшем будущем (до 2000 г.) большинство профессий будут испытывать влияние информационных технологий и станут использовать информационно-технические средства для решения своих проблем.

Ключевая роль информатики в обществе ясна. Поэтому она получит принципиальное значение для обучения в общеобразовательной школе. Необходимо подготовить учащихся к их будущей трудовой деятельности в обществе. Мы должны ориентироваться на определяющие факторы научно-технического прогресса, к которым относится и информатика. Введение основ информатики в школьное образование — требование времени, с которым мы должны считаться.

Нужен ли специальный предмет «Информатика» в общеобразовательной школе?

Введение дополнительного предмета «Информатика» по-моему не является необходимым, а в определенном смысле может быть даже вредным. Обязательный для всех учащихся, он имел бы следующие отрицательные аспекты. Дополнительный предмет «Информатика» означал бы по меньшей мере дополнительную нагрузку. Необходимое время занятий следовало бы получить за счет сокращения других предметов, для которых существуют свои проблемы с усвоением материала.

В таком случае общее образование испытало бы ненужное смещение акцентов. Всего этого можно избежать, если творчески отнестись к информатике и использовать ее возможности при изучении соответствующих предметов. Существование дополнительного предмета «Информатика» потребовало бы большого числа новых учителей-предметников по информатике, а их обучение может осуществляться лишь постепенно. Оно не обязательно должно происходить в таком объеме, если усвоение информатики будет распределено по всем предметам. Главную роль при этом должны и могут играть учителя математики и физики.

Существование отдельного предмета «Информатика» в школе может легко оказаться оправданием для других учителей, которые не имеют особого жела-

ния заниматься проблемой информатики в своих предметах. В этом случае им не нужно будет уделять внимание вопросам этого предмета, так как в школе есть специалисты по информатике, и им не нужно будет ориентировать методику своего предмета на передачу знаний — задача, которую в противном случае нельзя будет игнорировать.

Эти аргументы относятся и к обучению с использованием компьютеров в общеобразовательной школе, т. е. к самому широкому образованию. Они не направлены против введения специального курса информатики или углубленного предмета «Информатика» в специальных школах, где, наоборот, в преподавании, исследовании и воспитании следует уделять особое внимание формированию и определению обобщающей науки «Информатика» за счет подготовки научной молодежи и передачи специальных знаний по информатике в соответствии с потребностями предметов.

Рис. 1. Мини-ЭВМ, выпускаемая в ГДР, является программируемой. Она состоит из блока ввода с клавиатурой и процессора. В состав комплекса входят кассетный магнитофон и телевизор «Юность» в качестве устройства вывода.

Изменения в процессе обучения, вызванные использованием компьютеров

Применение персональных компьютеров с устройствами ввода — вывода и графическим дисплеем с достаточно высокой разрешающей способностью является для большинства предметов общеобразовательной школы задачей, которую постепенно должны решать все учителя-предметники. Использование ЭВМ в учебном процессе дает такие преимущества. Акценты в учебной работе смещаются от доминирующего фактического знания к процедурному. Усиливается роль проблемных задач. Значительно снижается доля (необходимой для ручных вычислений) рутинной работы в учебном процессе, которая уже и так сокращена благодаря введению школьных калькуляторов, что позволяет улучшить творческую работу учащихся.

Обучение становится более конкретным, так как правильность путей решения, например формул, проверяется в процессе обучения — в меру точности ЭВМ. (Обычно на числовое вопло-

щение результативного пути решения на уроке не остается достаточно времени, и учащийся должен выполнять вычисления позже, например в качестве домашней работы).

С помощью дисплея передача знаний становится более интенсивной, но по психологическим причинам следует остерегаться обилия впечатлений. Знания могут быть представлены в наглядной и комплексной форме графической (в том числе и цветовой) информацией (графики функций, геометрические фигуры, меню).

Особенно важным представляется то, каким образом могут быть использованы графические возможности ЭВМ, например, при решении задач о колебаниях пружины. Реальный объект в форме модели может быть представлен синхронно вместе с динамическими изменениями. Так, учащийся наблюдает колеблющуюся пружину в образе материальной модели синхронно с временной функцией ее отклонения или фазовой диаграммой изменения скорости на экране.

С помощью ЭВМ учителя и учащиеся (например, в рамках кабинетных занятий) могут проводить эксперименты. В результате может быть достигнута такая важная цель обучения, как интерес к объекту занятий и любовь к предмету.

Увлечение игрой присуще и взрослым и детям. Благодаря ЭВМ игры дополняют компьютерное обучение. Речь не идет о том, чтобы учащимся изменять определенные параметры программы и тем самым влиять на количественное решение задачи, а о том, чтобы дать возможность школьникам целенаправленно управлять графическим представлением на экране, решая задачи с помощью ЭВМ, чем чисто аналитическим путем. Ясно, что указанные цели могут быть достигнуты только путем программирования формул или процессов решений. Такие рамочные системы служат для активизации специализированных программ, используемых для занятий. По желанию эти программы модифицируются. Большое значение имеет управление использованием учебного программного обеспечения с помощью соответствующих меню, параллельное представление графической и алфавитно-цифровой информации, фактор динамики, разгрузка экрана от избыточной информации, а также возможность гибкого увеличения графической информации (принцип математического микроскопа).

Как могут осуществляться контакты учителей и учащихся с персональным компьютером?

Желательно, чтобы в ближайшем будущем в ГДР учителя-предметники общеобразовательной школы использовали ЭВМ для обучения. При этом нет необходимости изучать большой объем курса повышения квалификации и изучения информатики как дополнительного предмета. Школьники и педагоги могут легко привыкнуть к обращению с персональным компьютером, подобно тому как любой человек с нормальными способностями может научиться водить автомобиль.

В этом значительную помощь окажет простой в усвоении язык программирования высокого уровня. Например, Бейсик, хотя, как правило, в зависимости от его реализации на персональном компьютере (операционная система) язык не удовлетворяет некоторым потребностям пользователя. Преимущество Бейсика состоит в том, что он может быть изучен самостоятельно или непосредственно в процессе использования персонального компьютера на занятиях. Специальный курс Бейсика не является необходимым, однако может изучаться в сочетании с обучением математике. Однако не требуется вводить обязательный курс Бейсика как предмет в школе, так как небольшого числа команд Бейсика уже достаточно для того, чтобы иметь возможность разумно общаться с ЭВМ. Отсутствующие элементы легко освоить самостоятельно при возникновении новых потребностей в общении с ЭВМ.

Бейсик в сочетании с мощными операционными системами современных персональных компьютеров позволяет составлять прикладные программы и управлять работой ЭВМ.

Последовательное введение персонального компьютера в образование

В соответствии с растущими техническими возможностями ГДР по производству персональных компьютеров и уровнем подготовки учителей к использованию современных информационно-технических средств введение ЭВМ в общеобразовательной школе могло бы осуществляться так.

Сначала персональный компьютер вводится на уроках математики. При этом возрастает интенсивность сообщения знаний. В курсах Бейсика математические знания могут быть широко использованы для целей демонстрации. После введения персонального компьютера на уроках математики надо использовать возможности ЭВМ и в других предметах, возможно, в такой последовательности: физика, химия, введение в социалистическое производство и затем в остальных предметах. В гуманитарных предметах имеет смысл использовать ЭВМ лишь тогда, когда будут предоставлены специальным образом подготовленные запасы знаний, а школа сможет получать персональные компьютеры с соответствующим объемом памяти.

Для реализации обоих этапов сообщения знаний необходимо, чтобы соответствующий учитель-предметник имел на занятиях персональный компьютер. В быстром распространении этого процесса важно наличие оборудованных персональными компьютерами кабинетов для пользования ими во внеурочные часы, а также формирование обществ и кружков учащихся на факультативной основе. Они могут одновременно быть использованы и для разработки программного обеспечения для обучения.

Когда школы в достаточной мере будут оборудованы ЭВМ, учащиеся смогут самостоятельно или под руководством учителей работать с имеющимися в кабинетах средствами электронной обработки информации для выполнения целенаправленных экспериментов, решать с помощью ЭВМ свои домашние задания или выполнять работы, направленные на последующую продуктивную деятельность.

Когда персональные компьютеры поступят в продажу, как, например, телевизоры, эффект применения ЭВМ в учебном процессе может быть значительно усилен. Школьные задания уже не будут восприниматься учащимися как неприятная нагрузка, а как толчок к занятиям с компьютером дома.

Высшая ступень интеграции персонального компьютера в обучении будет достигнута, когда установленные в классе компьютеры будут объединены в сеть с компьютером учителя.

Учитель сможет благодаря приоритетному управлению запрашивать информацию от учащихся или передавать им информацию на экран. При этом технические средства обучения (ТСО), доска а также реальные эксперименты должны в значительной мере сохранить свое место.

Здесь не делается попытка, как это было при прежних формах программированного обучения, постепенно устранить учителя из учебного процесса, наоборот, его роль возрастает в общении с учащимися с поддержкой ЭВМ. Личное воздействие педагога не должно быть снижено. Учитель остается воспитателем и другом учащихся. Естественно, при этом изменятся отношения учитель — ученик: педагог будет стоять во главе учащихся, вместе с которыми он прорабатывает материал, используя все доступные технические средства.

Так, для занятий музыкой главное — настоящий музыкальный инструмент и владение этим инструментом учителем. Однако следует использовать богатые возможности электронного синтезатора музыки.

Нужны ли новые учебники при использовании ЭВМ в общеобразовательной школе?

Пришло время составления новых учебников, включающих в себя элементы использования вычислительной техники, причем независимо от того, насколько быстро может быть удовлетворена потребность в электронно-вычислительных средствах.

Рис. 2. Центральный учебный кабинет по информатике, оборудованный вычислительной техникой, работает с 30 сентября 1985 года в Университете им. Гумбольдта в Берлине.

Учебники в любом случае должны быть ориентированы на соответствующий процесс. Новые учебники могут быть построены примерно следующим образом. Учебники представляют материал в наиболее целесообразной последовательности, логически построенной и педагогически и содержательно продуманной. Каждый относительно замкнутый фрагмент знаний (как модуль), содержащий факты или знания о методах решения проблем, дополняется продуманными примерами, практическими упражнениями в сочетании с рисунками и данными реализованными из области соответствующих на ЭВМ алгоритмов решения проблем. К примерам и упражнениям принадлежат также относящиеся к тексту программы, сформулированные на языке программирования высокого уровня, преимущественно на Бейсике. Они могут быть легко активированы, содержат комментарии, предоставляют наглядные меню, снабжены демонстрационными примерами и представляют результаты хорошо подготовленной, по возможности гибко модифицируемой пользователем графики на экране.

Необязательно в явном виде включать в книги программы. Они могут быть записаны на кассете или дискете. Каждая программа обладает специальным содержанием и логикой решения. При этом соответствующая содержанию логика решения должна быть включена в текст книги. Упражнения и задачи, решаемые с помощью ЭВМ, необходимо ориентировать на глубокое проникновение в проблему предмета. Цель должна соответствовать содержательной разработке решения с помощью составления программы решения (содержание плюс логика решения), в дополнении и расширении содержащихся в учебнике программ.

Существует ли опасность при бесконтрольном использовании персональных компьютеров в школе и дома?

Всем пользователям должно быть указано на опасности, которые могут

возникнуть при неограниченном использовании персональных компьютеров. Так же как вредно смотреть все передачи по телевизору, так и целый день проводить за компьютером в молодом возрасте опасно.

Если теледопинг ведет к недостатку активности, компьютер стимулирует в высокой степени индивидуальные, творческие способности, что в определенных рамках следует весьма приветствовать. Но при избыточности это ведет к диспропорциям в развитии личности и может служить препятствием в формировании других способностей человека. Связанная с этим умственная перегрузка может вредно сказаться на здоровье.

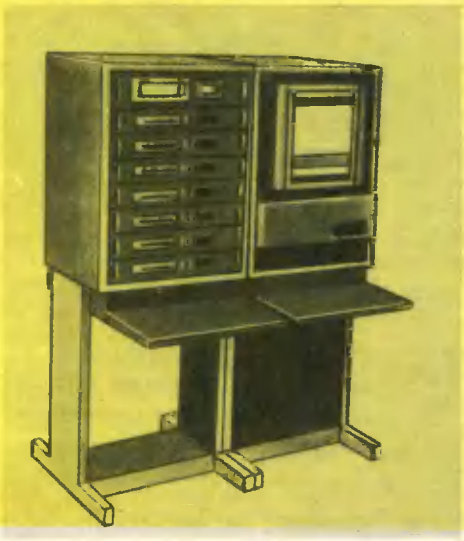
Преодолению этих опасностей в обучении способствует дух коллективизма, дружеская обстановка. При длительной работе за компьютером должны делаться паузы, во время которых учащийся может заниматься другим делом: спортом, играми, пением. Это правило надо соблюдать и при работе с компьютером в свободное время.

Еще одна обоснованная опасность заключается в склонности к привычке сводить всю полноту творческого решения проблем с использованием параллельного мышления и ассоциаций к чисто рекурсивным, последовательным процедурам решения. Против этой опасности может помочь обсуждение проблем в коллективе, проведение общих исследований различных вариантов с целью стимулирования идей, необходимых для решения задачи, и развития ассоциативного мышления. Эти, а возможно, и другие опасности нужно ясно осознавать и анализировать, чтобы иметь возможность соответствующим образом им противостоять.

Мы стоим в начале пути процесса введения компьютера в общеобразовательной школе, для того чтобы в короткие сроки решить поставленные проблемы на благо молодого поколения, на пользу социалистическому обществу.

Управляет ЭВМ

Производство фанеры требует точного соблюдения технологического режима, особенно при обработке древесины в специальных бассейнах, подогреваемых паром. Управлять этим процессом помогает ЭВМ. Советские конструкторы разработали оригинальную систему САРГ-1М (см. рис.), которая способна с высокой точностью контролировать технологию гидротермической обработки фанерного сырья. Датчики сообщают ЭВМ «Электроника-60» данные о начальной влажности и температуре древесины, поступающей в бассейн, о среднем диаметре деревянных заготовок. Компьютер, анализируя эту информацию, выбирает оптимальный технологический режим, при котором сырье приобретает заданные механические свойства. Кроме того, ЭВМ может рассчитывать и контролировать время прогрева древесины, регулировать температуру воды в 12 секциях бассейна, записывать ее на диаграммной ленте. При отклонении температуры от заданной включается световая сигнализация.



Испытания новой системы показали, что ее применение на деревообрабатывающих предприятиях уменьшает расход пара на 20—40 %, при этом выпуск высококачественной фанеры увеличивается. Применение ЭВМ на фанерной фабрике повышает производительность труда рабочих, улучшает качество выпускаемой продукции.

НЕВЕРОВА Д., ХОЗИЕВ В.

Дети о компьютерах

Введение в школьную программу такой дисциплины, как основы информатики, существенно изменяет соотношение других учебных дисциплин в школе. Ведь раскрытие содержания этого учебного предмета предполагает не только практическую деятельность с микрокалькулятором или овладение приемами программирования. Потребуется комплексная переработка основных учебных дисциплин, преодоление их разобщенности. Ряд ключевых понятий информатики, таких, как «моделирование», «алгоритм», «информация», могут и должны изучаться на примерах, взятых из математики, физики, химии, биологии и т. д., обеспечивая синтез научных «отдельностей» в мышлении школьника.

Следует также учесть, что понимание границ применимости формализованных и неформализованных методов научного познания, учет диалектического движения предмета анализа, понимание относительности и опосредованности полученного результата — все это может *прийти* из информатики и, будучи раскрыто в практической деятельности с конкретным материалом, способствовать формированию обобщенного знания о мире.

Наряду с решением вопросов о целях обучения и содержании данного учебного курса необходимо формирование у учащихся соответствующей мотивации и познавательного интереса к новому предмету.

Здесь трудно переоценить значение научно-популярной литературы и средств массовой информации. Но, к сожалению, они нередко способствуют распространению компьютерного мифа, атмосферы сенсационности, нежели становлению истинной компьютерной грамотности. Сведения, приобретаемые детьми вне школы, отрывочны и не систематизированы и обычно определяются не логикой теоретических знаний, а логикой повседневной жизни. Неверность и наивность иных представлений препятствуют развитию творческого мышления, способствуют формальному усвоению знаний, неумению связать теорию с практикой. Поэтому, на наш взгляд, необходимо исследовать уже сложившиеся вне школы и еще складывающиеся стихийные представления детей о мире техники и, в частности, о «компьютерном мире».

Таким образом, среди актуальных проблем компьютеризации обучения проблема стихийной компьютерной грамотности должна быть учтена, с тем чтобы не только вовремя скорректировать возможные негативные последствия такой грамотности, но и использовать ее для формирования интереса к новому учебному предмету.

Целью исследования (проведено в августе — декабре 1985 г. в Москве), предлагаемого читателям, являлось определение уровня осведомленности школьников различных возрастных групп в области техники, в частности вычислительной, и программирования; а также выяснение отношения детей к проблеме «Человек и ЭВМ».

Исследование заключалось в интервьюировании с последующим анализом и сравнением полученных данных. Интервью проводилось со школьниками в возрасте от 7 до 12 лет. Всего было опрошено 108 школьников со следующим распределением по возрастам: 7 лет — 20 детей, 8 лет — 15, 9 лет — 18, 10 лет — 20, 11 лет — 17, 12 лет — 18. Вопросы были разбиты на смысловые группы.

Осведомленность в области техники, владение основными техническими понятиями

Интересуется ли школьник вопросами тех-

ники? Какие ему известны типы и виды машин? Известно ли ему что-нибудь об устройстве названных машин?

Осведомленность в области вычислительной техники

Что такое ЭВМ? Как действует ЭВМ? Перечислить ее основные составные части. Пробовал ли школьник работать с микрокалькулятором?

Знание отраслей использования ЭВМ

Перечислить отрасли народного хозяйства, в которых широко используются ЭВМ. В каких отраслях, по мнению школьника, они еще необходимы? Как называется профессия человека, который работает с ЭВМ?

Знание технологии программирования

Что такое программирование? Зачем оно нужно? Что представляет собой процесс программирования? Как выглядит программа для ЭВМ? Что такое информация?

Человек и ЭВМ

Кто такой программист? Интересное ли занятие программирование на ЭВМ? С какого возраста можно обучать программированию? Возможно ли запрограммировать человека?

В процессе интервьюирования ответы школьников фиксировались (оригинальные мысли или спорные высказывания — особо). Надо отметить, что названные вопросы определяли лишь канву интервью. Беседа носила характер свободного рассуждения школьника. Экспериментатор стремился создать доверительную атмосферу, поощряя наиболее полные ответы, так что даже провокационный вопрос: «Какая разница между ЭВМ и компьютером?» — вызывал буквально «полет фантазии» школьников. Однако необходимо отметить, что и в этой доверительной обстановке не всегда удавалось получить полный, обоснованный ответ.

Анализ данных исследования по каждой возрастной группе и последующее межвозрастное сравнение результатов показали, что наибольшее количество правильных ответов дано школьниками 9 и 11 лет. Дети всех представленных возрастов лучше информированы по вопросам 1, 2, 5-й групп. Об отраслях использования ЭВМ и технологии программирования они имеют очень незначительные сведения.

Подведя итоги межвозрастного сравнения результатов по каждой группе вопросов, установили, что максимальный балл набран учащимися 9 и 10 лет, минимальный — 7—8-летними.

Как показало исследование, в возрастном отрезке 8—10 лет уровень знаний школьников по предложенным вопросам не меняется.

Особое внимание было уделено анализу материалов по вопросам «Человек и ЭВМ».

По мнению школьников 11, 12 лет обучать программированию на ЭВМ следует уже с 10 лет, т. е. они считают себя вполне осведомленными в теоретических вопросах программирования, способными приступить к практическим занятиям на ЭВМ, относятся к факту существования такой машины, как к вполне обыденному явлению. Примерно таких же взглядов придерживаются и 10-летние.

По мнению же детей 7—9 лет, обучаться программированию можно начинать только с 14—16 лет. В их представлении компьютер — машина необычайно сложная, в какой-то мере сверхъестественная. Иногда дети склонны приписывать ей человеческие качества.

Как уже отмечалось выше, нами особо фиксировались оригинальные мысли или спорные высказывания. Например, по мнению некоторых учащихся 10—12 лет, использование ЭВМ в различных областях народного хозяйства не приносит пользы, так как при этом «разучивается мыслить человек».

Ответы на группу вопросов «Человек и ЭВМ», на наш взгляд, самые интересные и показательные. Поэтому данная группа вопросов была расширена, проведено дополнительное интервьюирование.

Чтобы заинтересовать школьников темой интервью, некоторые вопросы представили в форме коротких рассказов с проблемными ситуациями.

Например: «В некотором царстве, в некотором государстве, в XXI космическом веке жил-был царь. Правил он своей страной 30 лет и 3 года. И не было у него никого ближе единственного сына, а также лучшего друга — персонального компьютера. Прошли годы, царь состарился и умер. Нужно было выбирать нового правителя. Перед смертью старый царь составил завещание, в котором указал, кто станет его преемником: сын или компьютер».

Школьнику необходимо ответить, кто же, по его мнению, станет царем, учитывая при этом, что править страной будет самый достойный, умный, образованный и сильный.

Детям предлагались также следующие вопросы.

Кто умнее: человек или компьютер?

Умеет ли компьютер мыслить? Может ли компьютер действовать без участия человека? Сможет ли компьютер, обученный правилам игры в шахматы, одержать победу над шахматистом-человеком?

Представим такую ситуацию: в одном из турниров клуба «Что? Где? Когда?» против команды телезрителей играют две команды: знатоки и компьютеры. У кого больше шансов на победу над телезрителями?

Не будет ли полезно вместо людей-космонавтов отправлять в космические полеты программируемые машины? Кто, люди или компьютеры, наиболее эффективно выполнят задачи исследования?

Кто сильнее: человек или робот?

Выгодна ли на заводах и фабриках при выполнении тяжелого физического труда замена человека роботом? Учащийся должен аргументировать свой ответ.

Если робот принял человеческий облик, по каким качествам можно отличить от него человека?

В каких, на ваш взгляд, видах спорта победу одержат люди, а в каких роботы? Ответ аргументировать, учитывая, что тот или иной вид спорта предполагает наличие определенных физических и психических качеств и на победу может рассчитывать спортсмен, в значительной степени обладающий этими качествами.

Сравнив ответы детей по возрастным группам, мы выяснили, что школьники 7, 8, 10 лет считают компьютер умнее человека. Противоположного мнения придерживаются дети 9 и 12 лет. Все дети единодушны во мнении, что робот физически сильнее человека.

К сожалению, приходится подчеркнуть, что большинство детей 8—10 лет признают *превосходство* компьютера над человеком.

Было особо отмечено несколько интересных высказываний, например: «Человек главнее и значительнее компьютера, так как машина не может управлять людьми»; «В любом виде деятельности компьютер может действовать самостоятельно при условии неусыпного контроля человека за его работой» (Маша, 12 лет); «В состязании эрудитов победу одержит команда людей, так как в отличие от компьютеров люди имеют возможность общаться, совместно обсуждать варианты ответов» (Наташа, 10 лет); «Полностью заменить людей ни в каком виде деятельности нельзя, так как тогда существование людей потеряет всякий смысл» (Дима, 9 лет).

Анализируя данные интервью, мы выявили своеобразную тенденцию дегуманизации решения проблемы «человек — техника» 8—10-летними школьниками. Конечно, по результатам проведенного исследования нельзя делать выводы о технократической направленности содержания начального образования, об эмпирическом способе мышления, формирующемся в начальной школе. Однако полученные данные заставляют, на наш взгляд, задуматься о возможных перспективах изучения новой дисциплины в школе. В частности, необходим активный поиск в направлении гуманизации курса информатики. Как ни *неожидан* этот вывод, следует признать, что освоение информатики с программирования, в любых его формах, без связи с другими учебными дисциплинами только усугубит нежелательные тенденции в развитии мировоззрения школьников. В конечном счете эти мировоззренческие издержки препятствуют пониманию учащимися значимости всего естественнонаучного цикла.

Достижение компьютерной грамотности возможно только в контексте общечеловеческой и научно-мировоззренческой грамотности. В будущем это, вероятно, потребует комплексной переработки материала всего школьного курса, с тем чтобы информатика не стала *одним из* учебных предметов, а органично включилась в систему школьного обучения и, выступая лишь как средство развития мышления, но не как его эквивалент, дополняла и обогащала в сознании учащихся научную картину мира.

ЗАЙДЕЛЬМАН Я.
г. Уфа

Начало положено, что дальше?

Сегодняшний учитель информатики — уже не тот напуганный и недоумевающий преподаватель, которого летом 1985 г. заставили изучать странный и непонятный предмет. Теперь у него за плечами курсы переподготовки по программе IX и X классов, двухлетний опыт работы. Ему помогают «Книга для учителя» и публикуемые во многих

изданиях методические разработки, он понимает содержание курса и хорошо ориентируется в программном материале.

Что же дальше? Следующий шаг — переход от понимания сути предлагаемого программой материала к пониманию причин его выбора. Необходимо осознать внутреннюю структуру курса, его межпредметные связи и связи с реальным миром.

Время экстренной помощи, заключавшейся в ответах на вопросы «чему учить?» и «как учить?», прошло, теперь можно оглянуться и попытаться ответить на более глубокие вопросы: «Почему так?» и «Как лучше?»

Прежде чем выбирать что-либо, необходимо определить критерии выбора, сформулировать цель. Итак, зачем вообще школе информатика? Вопрос этот обсуждается очень широко, сталкиваются самые разные мнения — от скептического «совсем не нужна» до прагматического «для подготовки программистов». Пытаясь определить цель обучения, многие забывают, что речь идет об общеобразовательном предмете массовой школы, и сводят все к получению навыков работы на конкретной ЭВМ и программирования на конкретном языке. Как следствие такого подхода возникают отрицание безмашинного курса и нападки на алгоритмический язык.

Истинные цели курса глубже и серьезнее. Попытаемся сформулировать их.

Общетеоретическая цель. Овладение алгоритмическим стилем мышления, который включает в себя умение четко сформулировать заданные условия и требуемые результаты, разбиение большой задачи на малые, поиск решения в виде последовательности действий, выбор которых может зависеть от конкретных условий.

Подчеркнем, что в этой формулировке нет ни слова ни об ЭВМ, ни о программировании. Алгоритмическое мышление в таком понимании применимо (и необходимо!) практически в любой сфере человеческой деятельности.

Практическая цель. Понимание принципиальных возможностей современных ЭВМ, представление о круге решаемых ими задач. Умение выделить «человеческую» и «машинную» части конкретной задачи, построение модели и формализация «машинной» части, доведение ее до уровня программы для ЭВМ.

Заметим, что и здесь программирование занимает подчиненное место. Главное — понять, что можно, а что нельзя запрограммировать, и формализовать выбранную для программирования задачу.

Итак, цели поставлены, теперь надо выбрать соответствующие им средства. И тут программирование заслуженно занимает ведущее место. Именно в нем наиболее полно проявляются алгоритмические закономерности, поэтому программирование становится основным средством овладения алгоритмическим мышлением.

Выбор сделан. Мышление — цель, программирование — средство. Но средство это нуждается в дальнейшем уточнении. Выбранное содержание необходимо облечь в конкретную форму. Форма в программировании — это язык. На каком языке записывать

составляемые в процессе обучения алгоритмы и программы? Исходя из поставленных целей, определим требования к языку.

1. Язык должен быть простым. Главная трудность должна заключаться в составлении алгоритма, а не в его записи (это требование немедленно исключает такие языки, как Лисп, Ассемблер, Си и язык программируемого микрокалькулятора).

2. Язык должен допускать естественное представление основных алгоритмических конструкций (отпали Фортран и Бейсик).

3. Язык должен быть универсальным, т. е. допускать естественную обработку различных величин (Алгол, Фортран, Паскаль не содержат достаточно развитых средств работы с литерными величинами).

Мне не кажутся обязательными такие требования, как распространенность языка или наличие эффективной реализации. Скорее наоборот, выбранный для массового обучения язык имеет хорошие шансы стать широко распространенным.

Таким образом, существующие языки программирования не пригодны для массового обучения. Для решения этой проблемы были созданы специальные языки: Робик и Рапира Г. А. Звенигородского, алгоритмический язык А. П. Ершова. По своим идеям да и по внешней форме эти языки достаточно близки, главное их отличие заключается в трактовке понятия типа величины: в алгоритмическом языке тип связан с именем, в Рапире — со значением. Оба подхода имеют свои достоинства и недостатки, рассмотрение которых не входит в наши задачи. В существующей программе алгоритмический язык предлагается в качестве основы обучения, а Рапира рекомендуется для изучения как язык программирования на ЭВМ. Видимо, с таким подходом можно согласиться.

Отметим попутно особенности алгоритмического языка, которые делают его особенно ценным для использования при обучении. Это прежде всего русская нотация и свободный синтаксис записи выражений, которые дают ученику возможность сосредоточиться не на записи алгоритма, а на его сути.

Основным средством достижения второй, практической цели обучения должна быть работа на ЭВМ с богатым программным обеспечением. В условиях безмашинного обучения, когда нет возможности для отработки навыков практического использования ЭВМ, вторая цель фактически сливается с первой и достигается теоретическим изучением различных путей применения ЭВМ и решением прикладных задач на основе изученного ранее теоретического материала.

Цель поставлена. Средства определены. Теперь можно переходить к выработке конкретного плана действий, т. е. определить методику преподавания курса. При этом методика стратегическая (выбор конкретного материала для изучения, разбиение его по темам, определение их порядка и объема) должна предшествовать тактической (разработкам по отдельным темам и урокам).

Попробуем с выработанных позиций оценить предлагаемое в учебном пособии и книге для чтения учителя содержание и поурочное планирование курса X класса.

Раздел I. Устройство ЭВМ. Цель этого раздела — продемонстрировать принципиальную возможность автоматического исполнения алгоритмов. Для этого приходится говорить о таких достаточно сложных вещах, как система команд, алгоритм работы процессора, кодирование, физические принципы работы ЭВМ. На изучение этих технических подробностей отводится 12 часов — треть годового объема!

Между тем этот материал имеет лишь косвенное отношение к поставленным целям. Разумеется, ученик должен знать, что ЭВМ — это не более чем очень сложное техническое устройство. Но должен ли он вникать в принципы его работы? Думаю, это необязательно. С точки зрения поставленных целей гораздо эффективнее был бы функциональный подход, т. е. обзор не внутренних технических решений, а внешних характеристик, которые можно было бы связать с потенциаль-

ными возможностями конкретных ЭВМ и кругом решаемых ими задач. Авторы учебного пособия, ставившие перед собой задачу готовить не программистов, а пользователей, предлагают «смотреть на ЭВМ глазами программиста». Необходимо проявить последовательность и посмотреть на ЭВМ глазами пользователя.

Изучение устройства ЭВМ должно дать возможность ответа на вопросы «На каком компьютере можно решить поставленную задачу?» и «Какие задачи можно решать на имеющемся компьютере?».

При этом более глубокое изучение устройства ЭВМ можно рекомендовать для факультативных и кружковых занятий.

Раздел II. Знакомство с программированием. Необходимость глубокого по сравнению с IX классом изучения алгоритмического языка не вызывает сомнений, но необходимо еще раз продумать, методически обосновать (а может быть, устранить) разбиение его на две части. Вообще, обсуждение последовательности и принципов изучения алгоритмического языка заслуживает отдельного серьезного разговора.

Изучение языка программирования тоже работает на поставленные цели, и дело не только в том, что знание языка позволяет решать на ЭВМ практические задачи (вторая цель курса). Изучение нового языка помогает осознать суть алгоритмизации, отделить языковую форму от алгоритмического содержания. Главная задача здесь — показать, как фундаментальные алгоритмические конструкции приобретают свойственную конкретному языку форму, сохраняя свое содержание.

Вернемся к вопросу о его выборе. Сегодня программа предоставляет учителю свободу, но большинство останавливается на Бейсике и Рапире — именно эти языки введены в учебное пособие, лишь по ним доступны методические разработки. Мне кажется, что описание конкретного языка вообще не должно входить в учебник. Вместо этого следует выпустить серию брошюр, содержащих ориентированные на школьников описания языков, причем не куцые, предлагаемые сегодня, а более полные, многоуровневые, чтобы по брошюре можно было не только изучать основы в минимальном объеме обязательной программы, но и освоить язык глубоко, используя его машинную реализацию.

Более подробное обсуждение различных языков — тема для отдельного разговора. Здесь же мне хотелось бы обратить внимание на опасность изучения Бейсика. Об этом много говорят, но Бейсик продолжает свое победное шествие. Пока не поздно, его необходимо остановить!

Перечислю его основные недостатки.

В Бейсике отсутствует естественное представление таких алгоритмических конструкций, как если-то—иначе и пока. Приведенные в учебнике команды IF—THEN—ELSE (усеченная форма если) и WHILE—WEND (пока) относятся к очень редкой версии Бейсика, не встречаются на имеющихся сегодня машинах (что стыдливо признается в книге для учителя).

Нет аналога вспомогательным алгоритмам. Конструкция не имеет таких важнейших свойств вспомогательных алгоритмов, как независимость имен, передача параметров и возможность организации библиотек. Таким образом, эта конструкция не является эффективным средством разбиения большой задачи на малые и сведения нерешенной задачи к решенным (важнейшие навыки алгоритмизации!). Других средств для этого Бейсик не предлагает.

Выбор имен в подавляющем большинстве версий Бейсика ограничен формулой «буква+цифра». Использование содержательных мнемонических имен запрещено, что очень затрудняет разработку и понимание программ, выходящих за рамки примитива.

Таким образом, Бейсик может быть инструментом программирования (хотя и очень несовершенным), но он не дает адекватных средств для развития мышления. В то же время из-за отсутствия естественного представления основных конструкций его очень неудобно использовать в качестве иллюстрации.

Особо опасно изучение Бейсика в

машинном курсе. Слабые ученики плохо понимают язык, путаются в многочисленных ОТО при программировании алгоритмических конструкций, теряют интерес к предмету и машине. На программирование у них уходит столько сил, что на мышление уже не остается.

Сильные же ученики, видя, что машина прекрасно понимает их без «алгоритмических излишеств» типа заголовка и продуманной структуры программы, пренебрегают этими деталями, не хотят писать на алгоритмическом языке задачи, которые можно сразу сделать на Бейсике. В итоге они получают навык программирования, а не мышления и главная цель остается не достигнутой.

Почему же, несмотря на столь серьезную опасность Бейсика, он так популярен? Перечислю основные аргументы, выдвигаемые его защитниками, и попытаюсь прокомментировать их.

Бейсик прост в изучении. Это — единственное достоинство языка как такового. Но это та самая простота, которая хуже воровства, «простота орудий каменного века», гибельное для боя отсутствие тяжести в ученье. Впрочем, в сторону эмоции. Бейсик прост, потому что в нем нет сложных элементов (в первую очередь вспомогательных алгоритмов и алгоритмических конструкций), без овладения которыми поставленные цели не достигаются.

При работе на Бейсике диалог с ЭВМ происходит быстро и просто. Это достоинства не языка, а его реализации. Пример Е-практикума и применяемые профессионалами символьные отладчики для языков высокого уровня показывают, что хороший диалог можно организовать на любом языке. Дело за программистами.

Бейсик имеется на многих машинах, он широко распространен. Это не достоинство, а большая беда. Фактически мы сталкиваемся здесь с осужденным в промышленности диктатом производителя. Изготовители ЭВМ и программного обеспечения рекламируют Бейсик, так как Бейсик-системы проще в изготовлении и могут быть перенесены с западных компьютеров. Пользователи приобретают Бейсик частью по незнанию истинного положения дел (это особенно характерно для школ и органов министерств просвещения), а грамотные люди часто вынуждены брать его, так как им не предлагают ничего другого.

Необходимо, чтобы поставляемые в школы компьютеры имели систему программирования на языке действительно высокого уровня. Остановить Бейсик — наша первоочередная задача.

Раздел III. *Роль ЭВМ в современном обществе. Перспективы развития вычислительной техники.* С точки зрения поставленных целей, этот раздел — один из важнейших в курсе. И двух часов на его изучение (экскурсии не в счет) конечно же мало. Поэтому я предлагаю следующее:

1. Объединить изучение разделов I и III. Помимо увеличения времени на рассмотрение применений ЭВМ это даст возможность прочнее связать устройство ЭВМ с их возможностями (об этом уже говорилось в обзоре раздела I).

2. При безмашинном курсе передать на эту тему часть часов, отведенных на язык программирования.

Есть и более радикальное предложение: перевести раздел II в IX класс, сосредоточив там полностью изучение алгоритмического языка и языка программирования. Помимо чисто методических выгод это облегчит летнюю практику на базовых предприятиях, которую организуют некоторые школы. При этом в X класс переходит тема «Решение задач с помощью ЭВМ». Таким образом, курс X класса обретает концептуальную целостность. Он целиком посвящается использованию ЭВМ и рассматривает соответствующие цели (назначение ЭВМ), средства (устройство ЭВМ) и методы (решение конкретных задач).

ЧИТАТЕЛЬ ПОЗДРАВЛЯЕТ, НАСТАИВАЕТ, ТРЕБУЕТ...

Здравствуйте, дорогие учителя, преподаватели, методисты и... ученики!

Да, представьте себе, среди наших подписчиков есть и те, кого мы учим и для кого, собственно, и затеяна огромнейшая компьютерная работа.

Мы внимательно прочитали все ваши письма.

Спасибо за слова поддержки и одобрения! «Я уже не представляю себе, как бы я готовился к урокам, если бы не «Информатика и образование»,— пишет учитель из Херсонской области Николай Тихонович Геспало.

«Печатайте больше задач!»— такова просьба преподавателей В. Медведева из Одессы, Л. Полупанова из Вильнюса и многих других. Уже в этом номере выполняем читательский заказ. Просим присылать и ваши собственные, пусть даже небольшие, методические разработки.

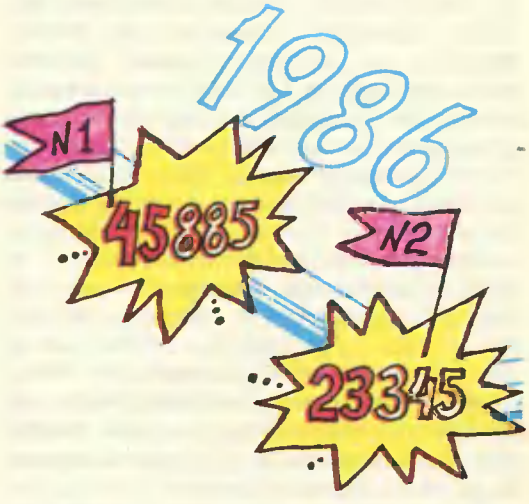
Сообщаем подписчикам, обратившимся с проблемными письмами, что по наиболее острым вопросам готовятся публикации с выездом сотрудников редакции на места.

Ребятам, которые подписались на «Информатику», предлагаем *участвовать* в работе журнала в журнале «Молодежная инициатива».

А вот многих и многих вынуждены огорчить — простите, но мы не можем выслать самые первые наши номера.

Группа молодых ученых из Москвы (они, кстати, являются авторами одной из статей этого номера), Д. Волков, П. Ширков и другие считают, что журнал должен помочь выработать коллективное мнение, какими путями будет развиваться школьная информатика. Согласны!

Хочу шутя отметить парадокс — на первых встречах с читателями мы убедились: многие знают Фортран, Бейсик,



Паскаль, но не все знают элементарное. «Оказывается, можно подписаться на журнал с любого номера!» — воскликнул Э. Ермаков (кстати, тоже один из авторов этого номера), когда сотрудники редакции встречались со студентами в МИЭМе. Удивительно, что при поголовной грамотности в нашей стране многие

товарищи пропускают «самые главные» строки из журнальных публикаций — «С ЛЮБОГО НОМЕРА».

Скоро лето. Говорят, что учителя, как и все люди, в отпуске стараются забыть о делах. А отпуск у педагогов заслуженно большой. Опытные сотрудники старейших педагогических журналов посоветовали нам попросить вас до летних каникул оформить подписку для школы, СПТУ, техникума, а в том, что вы лично уже читаете наш журнал,— мы не сомневаемся.

Мы с особенной любовью и тщательностью будем готовить номера, которые лягут к вам на стол в преддверии первого сентября.

Раскрою один редакционный секрет: мы намерены «внедрить» в одну из школ своего штатного сотрудника, который попытается окунуться во все мучения и радости преподавателя информатики.

Когда я придумывала сюжет рисунка, чтобы заказать художнику, может быть, юмористическое преувеличение насчет космических скоростей и было, а в остальном — никакой фантастики: тираж растет, читатели активны, редакция полна решимости делать полезный журнал.

Просим вас в письмах подписываться полным именем и отчеством. Нам грустно обращаться к своим единомышленникам по типу «Ув. тов. Иванов!».

С уважением

Татьяна Николаевна ДРАГНЫШ,
научный редактор

Ю. МОЛОТКОВ

Поможет совет

В Новосибирской области сложилась и развивается организационная система компьютеризации обучения в школах, СПТУ, средних специальных учебных заведениях и вузах.

При Новосибирском обкоме партии создан совет по научно-техническому прогрессу, главная функция которого — содействие развитию научно-технического прогресса на промышленных предприятиях и в организациях, соединение науки с практикой. Советом выработаны рекомедации по подготовке кадров во всех типах учебных заведений в соответствии с современными потребностями общества.

При совете по научно-техническому прогрессу образована секция информатики и вычислительной техники. Ее основные задачи таковы: выработка единого метода организации компьютерного всеобуча; подготовка и переподготовка кадров; организация методического обеспечения; создание регионального банка программного продукта; решение организационных вопросов по оптимальному использованию средств ВТ; решение межведомственных проблем.

Более частными задачами занимается методический совет, созданный при секции информатики и вычислительной техники. Главным образом, это решение методических и методологических вопросов, связанных с компьютеризацией обучения.

В состав методического совета вошли представители отдела науки и учебных заведений обкома КПСС; народного образования; высших учебных заведений; института усовершенствования учителей; средних специальных учебных заведений; профтехобразования; вновь открытого Института информатики и вычислительной техники.

Объединение таких сил позволяет выработать единую позицию в организации компьютерного всеобуча во всех типах учебных заведений; скоординировать возможности в подготовке кадров, в разработке обучающих программ; более рационально использовать имеющуюся вычислительную технику.

В 1985 г. при совете директоров средних специальных учебных заведений Новосибирской области была создана секция информатики и вычислительной техники.

За время существования секции подготовлено 130 преподавателей для чтения курса «Информатика и вычислительная техника». Проанализирован уровень оснащенности средних специальных учебных заведений средствами вычислительной техники. Создано областное методическое объединение преподавателей основ информатики и вычислительной техники. Разработано Положение об опорном учебном заведении, Положение о вычислительном центре при опорном учебном заведении. Проведен научно-методический семинар «Система применения технических средств обуче-

ния в методе эвристической беседы». В работе семинара приняли участие ученые Сибирского отделения АН СССР. Семинар проходил в терминальном классе техникума электронных приборов. Был опробован ряд обучающих программ. На базе вузов города проводятся научно-методические семинары, в работе которых участвуют преподаватели средних специальных учебных заведений.

На текущий учебный год методическим объединением преподавателей основ информатики и вычислительной техники было запланировано два семинара по обмену опытом и одна научно-практическая конференция по применению ЭВМ в учебном процессе.

Первый семинар по обмену опытом уже состоялся. На нем была представлена автоматизированная обучающая система «Барьер» как эффективное средство обучения программированию (автор Ю. Д. Емельянов — преподаватель техникума электронных приборов и др.).

В качестве образца разработан комплексный план внедрения информатики и вычислительной техники на примере среднего специального учебного заведения.

Определены три уровня внедрения:

Нулевой уровень — общий курс информатики и вычислительной техники.

1-й уровень — общий курс информатики и вычислительной техники плюс курс основ вычислительной техники.

2-й уровень — общий курс информатики и вычислительной техники плюс программирование, алгоритмические языки, курс основ вычислительной и микропроцессорной техники.

На совете директоров перед учебными заведениями была поставлена задача разработать комплексный план внедрения информатики и вычислительной техники и определить уровень обучения для каждой имеющейся в учебном заведении специальности.

Уже сегодня в учебных заведениях Новосибирской области разработаны и функционируют подсистемы «Успеваемость и посещаемость», «Автоматизированный контроль», которые позволяют вести с помощью ЭВМ оперативный контроль за исполнительской деятельностью коллектива. В техникуме электронных приборов внедрена обучающая система «Барьер»; ведутся расчеты с использованием ЭВМ по курсовому и дипломному проектированию в ряде техникумов области.

Опорные учебные заведения кооперируются в применении средств вычислительной техники с общеобразовательными школами, предприятиями, организациями и учреждениями, вузами.

В целом организационная работа, проводимая советом директоров и непосредственно секцией по информатике и вычислительной технике, координирует деятельность учебных заведений области по компьютеризации обучения.

Но вместе с тем имеются существенные проблемы, которые препятствуют компьютеризации обучения.

Оснащенность учебных заведений вычислительной техникой. На наш взгляд, целесообразно оснащать однотипной вычислительной техникой отдельные регионы страны, что существенно снизило бы затраты на разработку программного продукта.

Организация вычислительных центров при опорных учебных заведениях. Наша практика показывает, что средства вычислительной техники нужно прежде всего концентрировать в опорных учебных заведениях, где необходимы квалифицированные кадры программистов и обслуживающий персонал, что существенно повысит эффективность использования средств вычислительной техники.

Разработка программного продукта и обучающих программ. Как известно, разработчики неохотно делятся программным продуктом. Очевидно следует подумать о региональных банках программного продукта. К сожалению, ГОСТ на программный продукт не учитывает специфику программного продукта для обучения. Следует подумать о создании ГОСТа именно для обучающих программ.

Необходима, на наш взгляд, координация действий на уровне Минвуза СССР. Следует разработать план создания обучающих программ по регионам страны, а внутри каждого региона

с помощью совета директоров создать группы преподавателей, методистов и специалистов для разработки обучающих программ, что значительно ускорит процесс компьютеризации обучения.

Подготовка кадров. На наш взгляд, нужно подумать о создании региональных центров компьютерного всеобуча, которые бы не только осуществляли обучение, но и стали бы центрами научно-методического руководства, центрами накопления программного продукта.

Компьютеризация учебного процесса находится в стадии развития. Необходима координация наших действий, выработка стратегии развития. А практика проверит правильность выбранного пути.

Л. БУЛЫГИН, М. ЭГЛАЙС
ВЦ Латвийского государственного университета

Локальная сеть в школе

Оригинальная система локальной сети на микро-ЭВМ БК-0010 разработана в лаборатории проблем школьной информатики вычислительного центра ЛГУ им. П. Стучки. Испытания новой системы в школе № 20 Риги и Лиелвардской средней школе показали хорошие результаты. Работа в режиме локальной сети — практически единственная возможность эффективного использования ЭВМ во время уроков. Локальная сеть должна обеспечивать следующие режимы работы:

1. Загрузка учебных программ с внешнего носителя в компьютер учителя. Это могут быть обучающие программы и системы, используемые на уроках: программы, составленные самими школьниками.

2. Пересылка программ и данных по каналу связи от ЭВМ учителя к ученическим компьютерам, и наоборот.

3. Запись полученных от ученических и учительской ЭВМ программ на внешний носитель для считывания и использования на следующих уроках.

Выпускаемые промышленностью и поставляемые в школы комплексы учебной вычислительной техники, КУВТ «Ямаха», КУВТ-86 предоставляют эти возможности. Они заложены и в технических требованиях на разрабатываемые школьные ЭВМ следующего поколения. Однако в школы также поступают микро-ЭВМ различных типов. Например, компьютер «Электроника БК-0010». Необходимо было разработать систему, объединяющую в единую сеть ЭВМ, установленные в классе. При этом система должна быть простой и удобной в работе для учителя, проводящего урок; полностью совместимой с существующими и разрабатываемыми программными средствами, включая программы на языках Фокал, Бейсик, машинном языке, игровые интерпретаторы языка Бейсик различных версий, АОС «Рига» и т. д. Надо учитывать, что они должны воспроизводиться с помощью бытового кассетного магнитофона.

Нельзя переделывать или вскрывать корпус ЭВМ БК-0010, создавать дополнительную нагрузку для их цепей питания. Отключение отдельной ЭВМ не должно нарушить работоспособность всей системы. Ее простая аппаратная часть комплектуется только из радиодеталей и кабеля, которые можно купить в магазине, чтобы при желании любая школа могла подключать оборудование своими силами, при очень небольших затратах (порядка 10 руб. на рабочее место ученика).

В основе этой системы: аппаратная часть и сетевая программа, загружаемая в учительскую ЭВМ. Эта программа управляет всей работой системы. К учительскому компьютеру через коммутационное устройство подключа-

ется бытовой кассетный магнитофон для воспроизведения сетевой, учебных и прикладных программ, которые объединяются в пакеты. Пакет состоит из одной или нескольких программ, написанных на разных языках программирования и доступных при работе сети одновременно на всех ученических ЭВМ в режиме чтения. Содержание пакета определяется учителем, который может создавать, дополнять, изменять, объединять или делить пакеты, сохранять их на магнитной ленте. При этом учитель пользуется простыми средствами редактирования пакета, имеющимися в системе. В него можно включить и программы учащихся, полученные с учительских ЭВМ по локальной сети.

В оперативной памяти учительской ЭВМ находится один пакет, каждая программа которого доступна в режиме чтения из любого ученического компьютера. Работа на ученических ЭВМ ведется так, как будто программы читаются с помощью магнитофона. При этом скорость передачи данных примерно в 5 раз выше и нет сбоев и случаев несовпадения контрольных сумм. Аппаратная и программная реализация интерфейса кассетного магнитофона позволяет увеличить скорость передачи данных с 1,2 до 6 кБод.

Аппаратная часть сети состоит из двух согласованных коаксиальных кабелей для передачи программ. Необходимо лишь отделить постоянную составляющую сигнала конденсатором. Это позволяет загрузить 10—15 машин. Применение эмитерного повторителя на КТ-817 позволяет подключить практически неограниченное число ЭВМ. Например, сеть хорошо работала даже при нагрузке сопротивлением 5 Ом, что эквивалентно подключению 1000 машин (входное сопротивление интерфейса кассетного магнитофона 5 Ком).

Преимущества предлагаемой системы по сравнению с существующими вариантами локальной сети на КУВТ «Ямаха» и КУВТ-86 следующие.

Загрузка программ из учительской машины после запуска системы происходит постоянно без вмешательства учителя. В КУВТ «Ямаха» этот процесс требует совместных действий учителя и ученика, в КУВТ-86 проводится учителем со своей машины. Это для него намного сложнее, а отказы отдельных ученических машин могут полностью дезорганизовать работу системы.

Высокая надежность системы обеспечена взаимозаменяемостью учительской и ученических ЭВМ, высокой помехозащищенностью, малым количеством разъемов. Действия одного ученика не влияют на работу других машин. А в КУВТ «Ямаха» отсутствие соединения на одной ЭВМ приводит к отказу работы компьютеров дальше по линии.

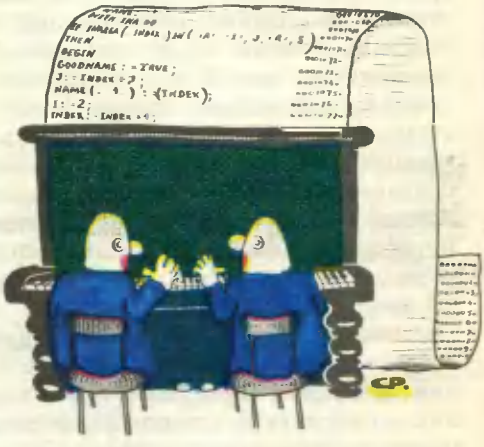
У системы сравнительно низкая стоимость и совместимость с существующим программным обеспечением. Работа по ученической ЭВМ та же, что и с отдельным кассетным магнитофоном. В стандартном режиме работы КУВТ «Ямаха» и КУВТ-86 не допускают пересылки программ на машинном коде.

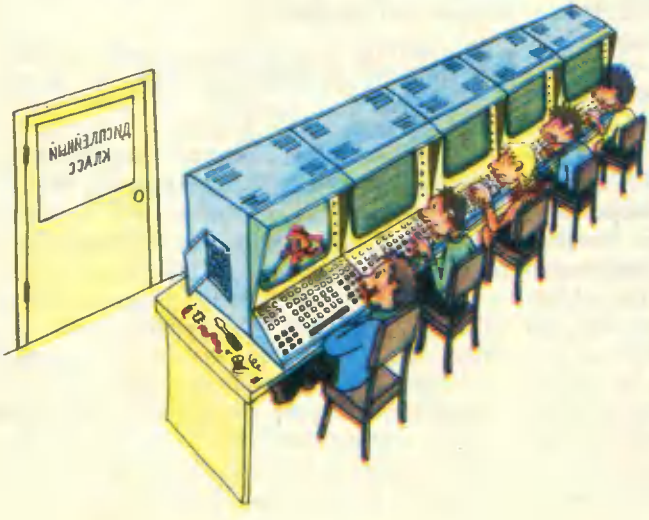
Скорость передачи данных учительской машины больше, чем у КУВТ «Ямаха» и ненамного меньше КУВТ-86. Максимальное время загрузки произвольной программы пропорционально размеру пакета и не превышает 1 минуты при максимальном пакете в 24 Кбайт.

В систему можно загружать в любой последовательности и с любого места программу из пакета.

Использование в классе однотипных ЭВМ не требует специального обучения для работы с ними, что особенно важно при нынешнем уровне подготовки учителей.

Веселый урок





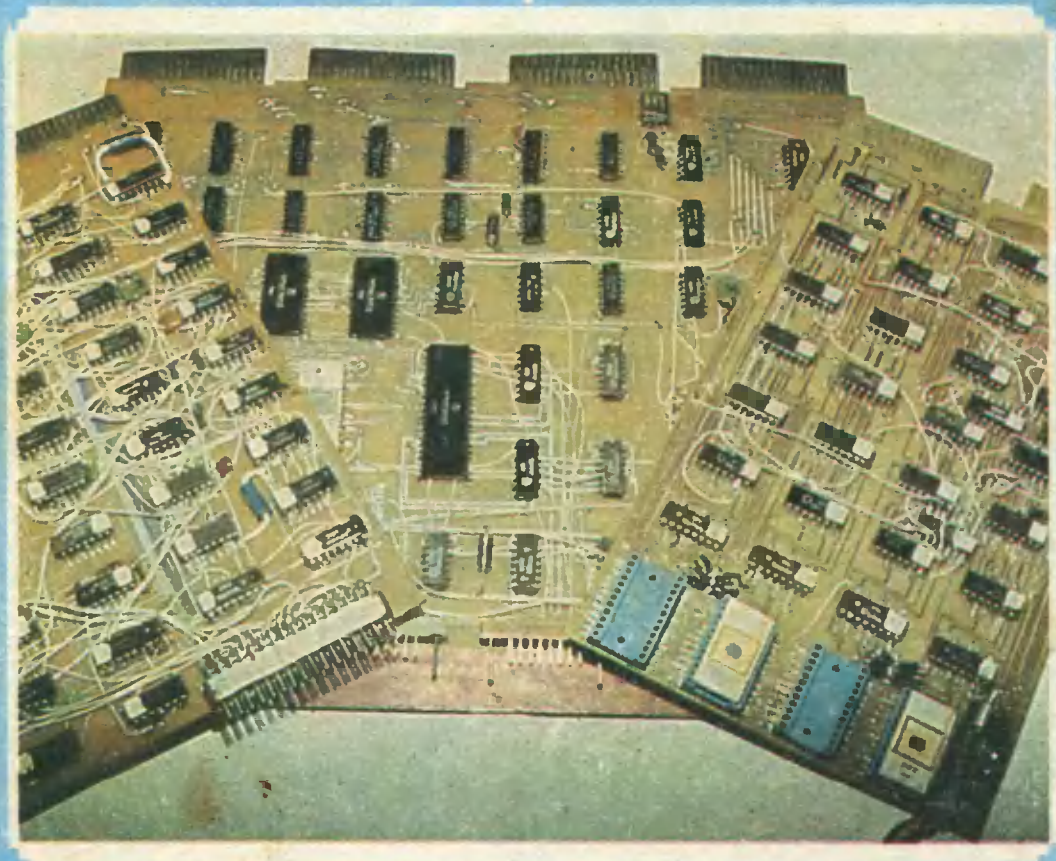
Цена 60 коп.

70423

ИНФО

3 '87

ИНФОРМАТИКА И ОБРАЗОВАНИЕ



Информатика и образование, 1987, № 3, 1—128