ISSN 0234—0453

ИНФО

ИНФОРМАТИКА И ОБРАЗОВАНИЕ

2 1992

Малое предприятие «Колледж» предлагает

Поставку класса «под ключ»

компьютерный класс УКНЦ или УКНЦ-01.01 (БК-0011М);
комплект столов под компьютеры;
пуско-наладочные работы;
установка сигнализации;
комплект программного обеспечения (курс информатики 102-116 часов);
обучение преподавателей.

Гарантийное обслуживание
Переналадку старых классов
Поставку программных разработок

■ **ESCORT** — автоматизированная обучающая система, способная в считанные часы научить вас самостоятельно работать на компьютерах типа ДВК, УКНЦ, «Электроника-85», БК-0011М, работающих в ОС RT-11.

В разделах АОС описываются:
внутреннее строение ЭВМ и периферийных устройств (клавиатура НГМД, принтер) и порядок работы с ними;
команды операционной системы;
прикладные программы и языки программирования;
работа с локальной сетью.

АОС ESCORT может быть полезна не только начинающим пользователям, но и преподавателям для проведения занятий по информатике. АОС имеет два режима работы: обучение и зачет.

■ **ИКС** — интерпретатор контролирующих систем. Позволит вам создавать контролирующие программы по любым учебным предметам. В тексте, выводимом на экран, могут использоваться до 33 специальных псевдографических символов. Программа работает на КУВТ-86 (БК-0010), УКНЦ-01.01 (БК-0011М).

■ **Колледж** — автоматизированный курс по информатике. Значительно облегчает преподавание основ информатики и вычислительной техники.

В комплект поставки входят: календарный план проведения занятий, пакет прикладных программ (автоматизированный курс информатики 92-146 часов), методические указания по проведению занятий.

Возможности программы:
два уровня обучения: простой и сложный;
два режима работы: обучение и зачет;
выставление оценок: за урок, тему, раздел, курс;
входной пароль с регистрацией.

«Колледж» успешно используется во многих учебных заведениях и может быть адаптирован под требования конкретного заказчика. Имеются версии для ДВК всех типов, УКНЦ, УКНЦ-01М, КУВТ-86.

■ **ANCAS** — программа группового копирования файлов и тиражирования кассет для КУВТ-86.

■ **АРМ** «Бухгалтерия» для IBM и ДВК. Позволяет компьютеризировать работу бухгалтерии учебного заведения.

Наш адрес: 107005, Москва, Волховский пер., 11.
Наш телефон: 261-66-87.

Научно-методический журнал.
Утвержден
Министерством образования РФ
и коллективом редакции

Издается с августа 1986 г.
Выходит шесть раз в год
Москва «Педагогика» 1992

ИНФО ИНФОРМАТИКА И ОБРАЗОВАНИЕ

2'92
март — апрель

Содержание

Общие вопросы

Методика обучения

Кабинет ВТ

Клуб «Корвет»

Клуб УКНЦ

Клуб БК

Педагогический опыт

Внеклассная работа

Молодежная инициатива

Информация

Обложка *С. Плюща*

Редактор отдела *А. Кравцова*
Научный редактор *Н. Копытина*
Зав. редакцией *Н. Игнатова*
Художественный редактор *Л. Коновалова*
Корректоры *М. Суворова, В. Антонова*

Сдано в набор 23.01.92. Подписано в печать 9.03.92. Формат 70×100 1/16. Бумага офсетная. Печать офсетная. Усл. печ. л. 10,40. Усл. кр.-отт. 42,88. Уч.-изд. л. 12,84 Тираж 56 620 экз. Заказ 51. Цена 2 руб.

Издательство «Педагогика»

Почту направлять по адресу: 119034, Москва, Смоленский б-р, д. 4. Издательство «Педагогика». Журнал «Информатика и образование».
Адрес редакции: Лефортовский пер., д. 8.
Телефон: 261-11-29.

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

В. КУРГАНСКИЙ

Виртуальные машины — концептуальная и методическая основа информатики

> Виртуальный объект (от ср.-век. лат. virtualis — способный) — идеальный объект, описанный в виде действующей программной модели, но проявляющий себя во взаимодействии с пользователем как реально существующий.
> Математический энциклопедический словарь. 1988. С. 813

Блеск и нищета информатики — один из парадоксов нашей действительности. Блеск — во впечатляющих приложениях, а нищета — в отсутствии общепринятых и конструктивных концепций, бесчисленных спорах о предмете, целях, задачах и других методических аспектах этой научно-технической и учебной дисциплины. Преодоление такой ситуации может быть облегчено и ускорено удачным выбором конструктивного концептуального ядра информатики. По мнению автора, в качестве такого ядра естественно использовать понятие виртуальной (абстрактной) машины.

История информатики и смежных с ней научно-технических дисциплин изобилует примерами применения абстрактных машин. Это и машина Неймана, и машины Тьюринга, и системы Поста, и конечные автоматы, и, наконец, виртуальные машины, а также другие псевдо- (или квази-) устройства (например, диски) в ряде операционных систем.

Виртуальная машина как концептуальное ядро информатики представляет собой совокупность ее основных понятий. Эта совокупность предназначена для точного и недвусмысленного представления различных концепций, ясного и конструктивно верного описания конкретных программно-технических средств. Другими словами, концептуальное ядро информатики в совокупности с некоторыми методическими принципами и приемами должно гарантировать: взаимную непротиворечивость представления программно-технического средства для пользователей, эксплуатационного персонала и разработчиков; сочетание полноты, ясности и краткости описаний различных аспектов изучаемых в информатике объектов, процессов и систем.

Ниже вводится концепция виртуальной машины (п. 1). В п. 2 в рамках этой концепции предлагается схематичное представление Бейсик-машины и окна экранного редактора текстов. Выбор примеров не случаен: Бейсик — одно из наиболее распространенных в практике обучения современных программно-технических средств, а окно — один из наиболее распространенных механизмов, с которым приходится иметь дело на персональных ЭВМ. Методические аспекты применения концепции виртуальной машины обсуждаются в п. 3.

1. Концепция виртуальной машины. Виртуальная машина — составное понятие, предназначенное для описания технических, программно-технических и абстрактных объектов, используемых в теории и практике современной информатики. Основными компонентами виртуальной машины являются: исполнитель; операционное пространство; программная память; устройства связи исполнителя с поручителем; параметры конфигурации и управления. Рассмотрим их подробнее. За основу примем соответствующие определения [1, с. 805—846]. При этом некоторые понятия будут уточнены или введены дополнительно.

Под исполнителем понимается абстрактное, техническое или программно-техническое устройство, способное выполнять команды из фиксированного набора. Принципиальное отличие виртуальных машин как объектов информатики заключается в том, что всякая команда, выполняемая исполнителем, оперирует значениями данных; результатом ее работы также являются значения данных. Значения данных размещаются в операционном пространстве. Элементы операционного пространства характеризуются именем (ад-

ресом) и значением. В набор простейших команд, явно или неявно реализуемых исполнителем, входят команды создания/удаления элемента операционного пространства и считывания/записи соответствующего значения. Таким образом, выполнение всякой команды приводит к изменению одного или нескольких объектов, принадлежащих операционному пространству. Эти изменения параметрически зависят от состояния множества объектов операционного пространства, называемых аргументами, или операндами, команды. Следует отметить, что всякая команда из системы команд является детерминированной. Это означает, что при одних и тех же значениях аргументов при выполнении данной команды всегда должны получаться одни и те же значения результатов.

Исполнитель может работать в командном режиме или выполнять заданную программу. В командном режиме он всегда находится в одном из двух состояний — ожидания команды или ее выполнения. В состоянии ожидания команды исполнителю может быть дана очередная команда, после выполнения которой он перейдет в состояние ожидания. При выполнении команды доступ к исполнителю заблокирован.

Из команд составляются программы. Текстуально программа — последовательность команд. При выполнении исполнителем программы его функционирование задается специальным алгоритмом. Для разновидности исполнителей — процессоров ЭВМ — этот алгоритм в учебной литературе называется основным. Выполняемая исполнителем программа должна быть помещена в специальное устройство — программную память. Алгоритм работы исполнителя по программе предназначен для определения последовательности выполнения команд, записанных в программную память и составляющих программу. Предполагается, что последовательность выполнения команд по программе не всегда соответствует их физическому или текстуальному размещению в программной памяти.

Программная память, как и операционное пространство, состоит из элементов, характеризующихся именем (адресом) и значением. Значение элемента программной памяти является командой. Программная память в некоторых случаях одновременно является и операционным пространством.

При работе с современными вычислительными системами редко удается иметь дело с одной виртуальной машиной. Этой привилегией, похоже, пользуются лишь посетители салонов компьютерных игр. Вычислительная система представляет собой совокупность взаимодействующих виртуальных машин. Базис этой совокупности машин составляют аппаратура ЭВМ, операционная система и штатные программные средства. Запуск программы под управлением операционной системы — это «рождение» новой виртуальной машины, а завершение исполнения программы — прекращение существования виртуальной машины. Параметры конфигурации и управления задают режимы работы и настраиваемые характеристики виртуальной машины, обеспечивают ее привязку к тем или иным компонентам базовых виртуальных машин. Например, в Бейсик-машине в качестве таких параметров выступают: умалчиваемые имена устройств и файлов для команд ввода-вывода, размер страницы на печатающем устройстве и т. д. В диалоговых машинах параметры конфигурации и управления являются, как правило, компонентами их операционных пространств.

Работа исполнителя в составе виртуальной машины инициируется поручителем (человеком или исполнителем другой виртуальной машины). Результаты работы исполнителя должны быть доступны поручителю. Взаимодействие поручителя и исполнителя осуществляется с помощью устройств связи. В настоящее время в диалоговых системах, где поручителем является человек, в качестве таких устройств связи широко применяются клавиатура и дисплей.

Рассмотрим возможные уровни взаимодействия поручителя и исполнителя. На низшем уровне программная память, если она имеется, а также часть операционного пространства с входными данными и значения параметров управления задаются не зависимыми от исполнителя средствами. Результаты работы исполнителя должны быть отображены на устройствах, обеспечивающих их последующее восприятие поручителем. Взаимодействие при этом сводится к инициации исполнителя на жестко или параметрически заданных размещениях программной памяти, операционного пространства и т. д. и получении сигнала о завершении работы исполнителя.

Высший уровень взаимодействия возможен, если исполнитель допускает командный режим работы. При этом в систему его команд должны входить команды редактирования параметров конфигурации и управления, а в случае возможности работы исполнителя по программе — команды редактирования программной памяти и инициирования программы. При этом устройства связи поручителя с исполнителем также могут выступать в качестве компонентов операционного пространства. Например, в Бейсик-машине клавиатура является устройством для кодирования и передачи исполни-

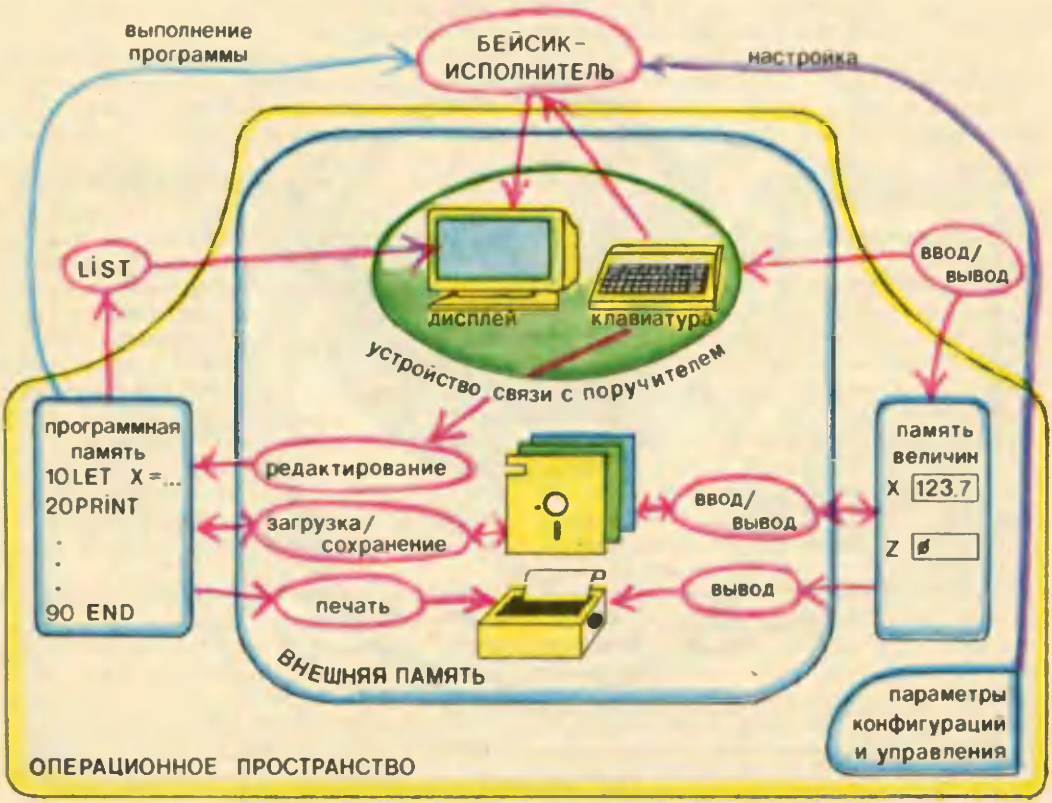
телю отдельных команд, а также одним из внешних устройств для ввода значений переменных. На практике применяются промежуточные по сравнению с описанными уровни взаимодействия поручителя и исполнителя.

Большая часть применений вычислительных систем сводится к последовательному использованию нескольких виртуальных машин, эксплуатируемых под управлением особой виртуальной машины — операционной системы. Пример типичной цепочки: редактор текстов, компилятор, редактор связей (компоновщик) и т. д. В особо сложных случаях, например при разработке операционной системы или в вычислительной сети, приходится иметь дело с совокупностью взаимосвязанных и синхронно или асинхронно работающих виртуальных машин. Операционные пространства таких машин пересекаются.

Значительную часть содержания информатики как учебной дисциплины применительно к конкретным программно-техническим средствам в рамках концепции виртуальной машины составляют: назначение и общее устройство программно-технического средства; общая схема функционирования программно-технического средства и алгоритм его работы по программе; алгоритмы выполнения команд, составляющих систему команд исполнителя, и устройство отдельных компонентов программно-технического средства. Отметим, что концепция виртуальной машины обеспечивает последовательное и систематическое изложение этих вопросов по единой схеме в рамках унифицированной терминологии.

2. Примеры виртуальных машин. Ниже предлагается схематичное представление диалоговой системы программирования Бейсик и окна в виде виртульных машин. На примере Бейсика демонстрируются возможности концепции виртуальных машин на макроуровне, а на примере окна — на уровне отдельных механизмов и команд. Описание других типов современных программно-технических средств: операционных систем, электронных таблиц, экранных редакторов текстов, программируемого микрокалькулятора и т. д.— в виде виртуальных машин с различной степенью детализации не затруднено никакими принципиальными соображениями.

2.1. Виртуальная Бейсик-машина — комплекс программно-технических средств, предназначенный для подготовки и эксплуатации программ, написанных на Бейсике. Основны-



ми ее компонентами являются (рис. 1): исполнитель (Бейсик-интерпретатор), программная память (исполняемая последовательность пронумерованных Бейсик-команд), память величин (скалярные переменные, массивы и константы Бейсика), внешняя дисковая память (программные файлы и файлы данных), параметры конфигурации и управления, печатающее устройство, дисплей и клавиатура.

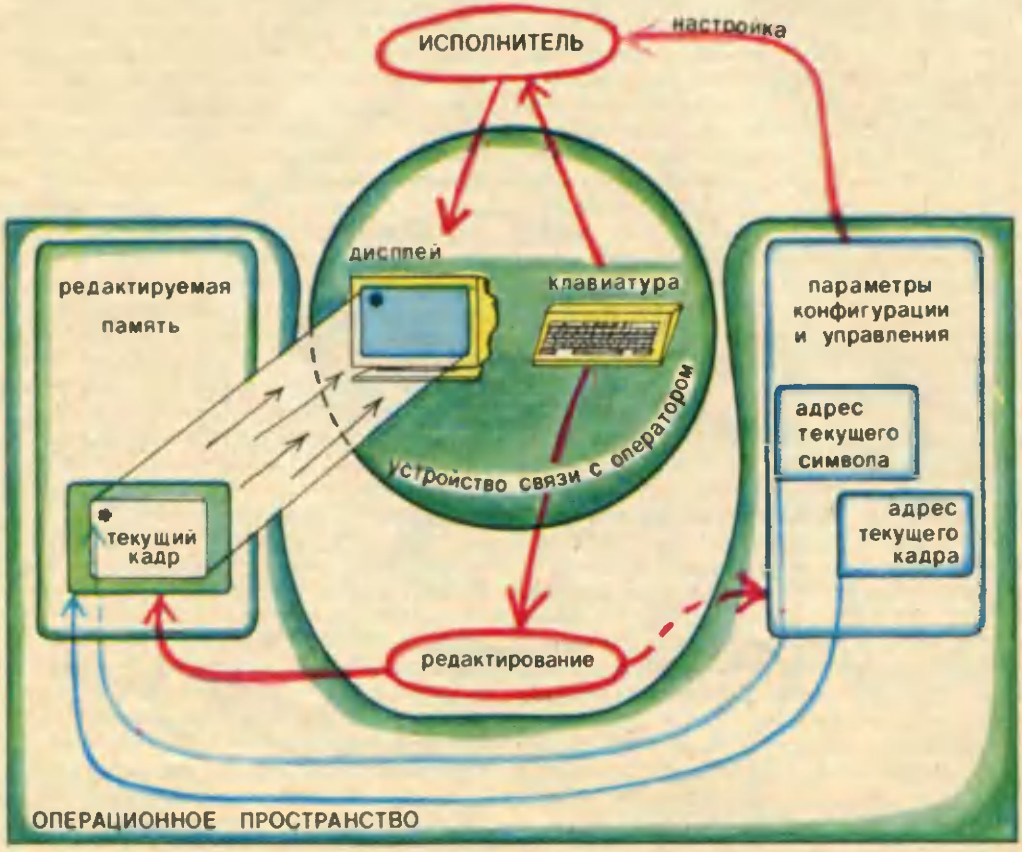
На рис. 1 одинарными и двойными заштрихованными стрелками показаны связи по управлению, а двойными стрелками без штриховки — связи по данным. Связи по управлению имеют место между любым компонентом Бейсик-машины и ее исполнителем. На рис. 1 выделены наиболее важные из них, а именно: связи исполнителя с программной памятью, параметрами конфигурации и управления, а также дисплеем и клавиатурой.

Связи по данным между компонентами Бейсик-машины на этом рисунке помечены обозначениями реализующих их вычислительных процессов, механизмов или команд управления этими механизмами: обмен программами между программой и внешней памятью, ввод-вывод (обмен между памятью величин и внешней памятью), редактирование программной памяти с клавиатуры, ее просмотр на экране дисплея и т. д.

Особенность виртуальной Бейсик-машины, как и других диалоговых систем программирования, заключается в том, что ее операционное пространство составляют: программная память, память величин, внешняя дисковая память, параметры конфигурации и управления, дисплей и клавиатура.

Общая схема работы Бейсик-машины включает ожидание/выполнение команд, задаваемых с клавиатуры, и выполнение команд, записанных в программную память. Знание общего устройства конкретного программно-технического средства в рамках концепции виртуальной машины, общей схемы ее работы, алгоритмов выполнения программы и команд составляет основную часть содержания информатики как учебной дисциплины. Детальное рассмотрение этих аспектов для Бейсик-машины не входит в задачи настоящей работы.

2.2. Окно как виртуальная машина (механизм). Окно — один из наиболее распространенных механизмов в диалоговых машинах. Общее устройство окна как виртуальной



машины приведено на рис. 2. Центральным компонентом операционного пространства окна является редактируемая память, которая представляет собой последовательность строк. Часть редактируемой памяти отображается на экране дисплея. Непосредственно для редактирования доступен только один из символов редактируемой памяти. Он называется текущим. Строка, в которой находится текущий символ, тоже называется текущей. Всякое окно устроено так, что текущий символ входит в отображаемый на экране кадр редактируемой памяти, который также называется текущим. Адреса текущего символа, текущего кадра, размеры и расположение на экране отображаемого текущего кадра редактируемой памяти и т. д. являются параметрами конфигурации и управления окна. Традиционно в качестве устройств связи с поручителем применительно к окну используются дисплей и клавиатура. В качестве дополнительных или альтернативных входных устройств связи могут использоваться «мышь» или другие подходящие средства.

Команды окна как виртуальной машины очевидны. Это изменение адресов текущих строки и кадра, изменение значения текущего символа, изменение значения текущего символа и всех следующих за ним символов текущей строки и т. д.

В составе Бейсик-машины вариант окна используется для просмотра содержимого программной памяти. Этот вариант отличается от полноценного механизма окна отсутствием возможностей изменения редактируемой памяти и командами изменения адреса текущего кадра.

В некоторых диалоговых системах (например, экранный редактор текстов ЛЕКСИКОН) из команд работы с окнами могут составляться программы. Эти программы в рамках концепции виртуальной машины принципиально ничем не отличаются от Бейсик-программ, Паскаль-программ или программ на машинном языке какой-нибудь ЭВМ.

3. Методические аспекты применения концепции виртуальной машины. Ниже обсуждаются: предмет информатики как научно-технической и учебной дисциплины, цели и задачи информатики как общеобразовательной дисциплины, а также некоторые методические аспекты обучения информатике.

Одним из центральных моментов настоящей работы является предложение считать предметом информатики виртуальные машины и их конкретные реализации. Это предложение базируется на тезисе о представимости любого программно-технического средства в виде виртуальной машины. Разумеется, этот тезис доказать нельзя. Можно доказать, что опровержение этого тезиса равносильно опровержению тезиса Тьюринга.

Цели и задачи информатики как общеобразовательной дисциплины в этом случае сводятся к следующим пунктам:

изучение концепции виртуальной машины как таковой;

практическое обучение в рамках концепции виртуальной машины работе с несколькими программно-техническими средствами (в том числе и программируемыми);

самостоятельное изучение одного или нескольких программно-технических средств на основе концепции виртуальной машины.

Состав изучаемых и самостоятельно осваиваемых программно-технических средств определяется общественными потребностями, имеющимися в наличии компьютерными средствами и индивидуальными особенностями учащихся.

Рассмотрим возможности концепции виртуальной машины, позволяющие преодолеть отсутствие и неоднозначность терминологии, сложность изучаемого предмета и ограниченные человеческие возможности восприятия, освоения и применения большого количества трудно конкретизируемых понятий информатики.

Что касается терминологических трудностей, то в силу универсальности концепции виртуальной машины в ее рамках мы имеем базис и правила построения всех необходимых (в том числе и вновь возникающих) терминов. При этом предполагается, что приведенная выше концепция должна развиваться и совершенствоваться, впитывая новейшие достижения философии, информатики и дидактики.

Проблемы сложности изучаемых объектов, процессов и систем, а также ограниченности человеческих возможностей, по-видимому, можно преодолеть, следуя (в частности, в плане представления учебного материала) концепции В. Ф. Шаталова [2]. Предполагается, что в рамках концепции виртуальной машины возможно создание унифицированных опорных сигналов, достаточно просто адаптируемых к конкретным целям, стратегии и тактике обучения, используемым программно-техническим средствам и другим особенностям педагогической деятельности. Следует отметить, что эта унификация не сделает процесс обучения однообразным и скучным. Универсальность концепции виртуальной машины проявляется в организации учебной работы и может быть заменена учащимися (вернее, продемонстрирована им) лишь на заключительных этапах обучения при самостоятельном освоении новых программно-технических средств.

В заключение отметим, что концепция виртуальной машины пригодна для обучения различных категорий учащихся, осваивающих информатику с различными целями. Это обеспечивается применением единого концептуального ядра при обучении и выделением/маскировкой конструктивных, эксплуатационных или пользовательских деталей конкретной машины обработки данных в зависимости от целей обучения.

Литература

1. Словарь школьной информатики // Математический энциклопедический словарь. М., 1988.
2. *Шаталов В. Ф.* Куда и как исчезли тройки. М.: Педагогика, 1979.

Подобно тому, как любой универсальный язык программирования пригоден для написания любой (в принципе) программы, так и предлагаемый понятийный аппарат, будучи универсальным, позволяет поддерживать любой курс информатики. Но, как и в ситуации с языками, его пригодность различна для различных курсов.

Нет задачи создания понятийного аппарата самого по себе, а есть задача построения курса со своими целями, понятийным аппаратом, содержанием и методикой преподавания. Можно предположить, что предложенный подход был бы неплох для курса с аппаратной, инженерной ориентацией; он может оправдать себя и в курсе программистского типа. Однако для пользовательско-ознакомительного курса его пригодность сомнительна.

В. БЕЛОШАПКА

Н. УГРИНОВИЧ

Телекоммуникационная сеть для системы образования

Современный этап информатизации общества характеризуется переходом от использования компьютеров в автономном режиме к созданию информационных сетей на базе компьютеров. В мире существует довольно много различных информационных сетей, каждая из которых объединяет большое количество пользователей, интересующихся определенной проблематикой. Особое значение за рубежом уделяется информационным сетям в сфере образования.

В нашей стране проблеме создания информационных сетей до сих пор уделялось недостаточное внимание, что сдерживает процесс информатизации образования и внедрение новых педагогических технологий в системе народного образования. Особенно это чувствуется сейчас, когда стремительно меняется содержание и формы образования, возникают новые типы учебных заведений (лицеи, гимназии), изменяется содержание и методика преподавания многих предметов (истории, литературы и т. д.).

Традиционная система распространения методических и нормативно-методических материалов, авторских разработок учителей-новаторов через педагогическую прессу, предметные журналы и отдельные издания слишком централизована, подвержена ограничениям в содержании и недостаточно оперативна. Система повышения квалификации учителей в институтах усовершенствования частично восполняет этот пробел, но требует от учителей больших затрат времени и далеко не всегда снабжает их необходимыми печатными и другими материалами.

Особенно ярко противоречие между содержанием и способом его распространения проявляется в области информатизации образования. Существующая

централизованная система тиражирования программного обеспечения через систему учколлекторов недостаточно оперативна, а предлагаемое школам программное обеспечение различными коммерческими организациями слишком дорого и часто не имеет методической ценности. Кроме того, большинство пользователей таких продуктов и разработок лишены дальнейшей поддержки и общения с авторами.

В традиционной системе информационного обмена упускается ученик, который часто является соавтором или автором программных средств. Творческий потенциал учащихся обычно ограничивается рамками одной школы, что, с одной стороны, обедняет рынок программных средств, а с другой — не стимулирует развитие творческой личности учащегося. Общение учащихся различных учреждений народного образования ограничено.

Все вышеперечисленное, а также созданная в системе народного образования материальная база, включающая десятки тысяч дисплейных классов и сотни тысяч компьютеров, создает объективные возможности создания информационной телекоммуникационной сети в сфере народного образования.

1. Организационная структура компьютерной телекоммуникационной сети SCHOOLNET

Предлагаемый ниже проект создания компьютерной телекоммуникационной сети SCHOOLNET позволит перейти на качественно новый уровень обмена информацией между учреждениями народного образования. Целью такой сети будет распространение нормативно-методической и методической литературы, программного обеспечения и обеспечение возможности информационного общения учителей и учащихся по интересующим их вопросам.

Компьютерная телекоммуникационная сеть образуется соединением компьютеров различных типов с помощью быстродействующих телефонных, спутниковых и иных линий связи в определенную иерархическую структуру. Очевидно, что обмен информацией в такой сети должен быть упорядочен, для этого у каждого абонента должен существовать свой компьютерный сетевой адрес.

Наиболее универсальный вариант информационного обмена, обмен сообщениями между двумя абонентами, реализуется с помощью компьютерной электронной почты, которая отличается от обычной тем, что доставка корреспонденции осуществляется по каналам электросвязи под управлением компьютеров. С ее помощью письмо, т. е. текст, снабженный стандартным заголовком (конвертом), доставляется по указанному адресу, который определяет местонахождение компьютера и имя адресата. Полученное сообщение (письмо) помещается в специальный файл, называемый почтовым ящиком, из которого адресат может извлечь его в любое время.

Если тема «разговора» интересна для многих абонентов, то эта тема фиксируется и выделяется в отдельную область. Далее составляется список участников дискуссии по принципу подписки и все сообщения, посланные абонентами на данную тему, рассылаются всем подписавшимся на тему. Обычно у такой темы есть координатор, который включает и исключает абонентов из группы, отслеживает процесс дискуссии и т. д.

При участии в дискуссии большого количества абонентов используется форма обмена информацией, которая называется телеконференцией. Эта система содержит в себе черты «доски объявлений» и «газеты». Все сообщения разделяются на темы или разделы, причем послать сообщение в любую тему может каждый абонент, получает же он все сообщения по интересующей его теме (или темам). Посылая сообщение, абонент указывает, к какой теме оно относится, и все, кто подписался на эту тему, данное сообщение получают. Отличие этой формы информационного обмена от предыдущей — в существенно более открытом доступе к информации для всех абонентов сети.

Наконец, в центре телекоммуникационной сети создается информационно-справочная система (банк данных) и любой абонент может выйти с запросом, найти и получить необходимую ему ин-

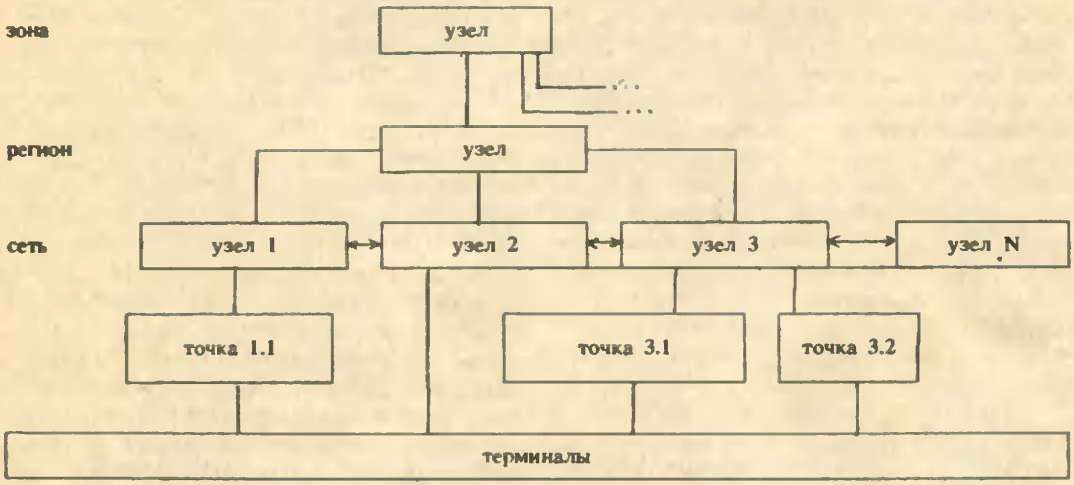
формацию или предоставить банку свою.

Для реализации вышеперечисленных принципов обмена информацией в структуре компьютерной телекоммуникационной сети целесообразно выделить три уровня, три типа структурных единиц: *узел, точка* и *терминал.*

Узел сети — это персональный компьютер, подключенный с помощью специальной аппаратуры к линии связи и имеющий необходимое программное обеспечение. В узле сети накапливается, хранится и рассылается по запросам абонентов информация. Узел связан с другими узлами и обменивается с ними информацией в заранее запрограммированном, автоматическом режиме (пересылает почту, отслеживает телеконференции и т. д.). Необходимым условием успешной работы узла является наличие специально подготовленного системного оператора.

Каждому узлу присваивается определенный сетевой адрес, который выполняет функцию почтового адреса и однозначно определяет местонахождение данного узла. Принят определенный формат сетевого адреса, который состоит из имени узла и адреса машины. Имя узла может быть произвольным, а адрес компьютера определяется самой структурой сети.

Узлы телекоммуникационной сети образуют территориальные структуры: сети, регионы, зоны. Сеть может включать в себя 10—20 узлов в пределах города или области. Регион — это автономная республика или другое территориальное образование. Зона — это еще более крупное объединение. Каждый из уровней территориальных структур имеет свой номер, и, таким образом, номер каждого узла формируется из совокупности номеров территориальных иерархических структур, к которым данный узел принадлежит. Например:



Для организации информационных обменов в сети необходимо иметь полный список ее абонентов, в котором в определенном формате записана информация о существующих узлах. Этот список постоянно обновляется и автоматически рассылается всем узлам, что позволяет легко определить координаты искомого абонента и установить с ним связь.

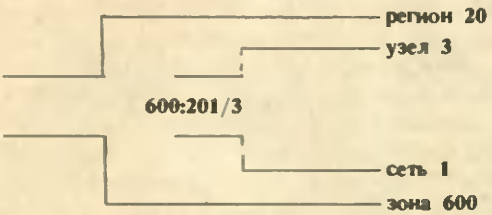
Между узлами организуются горизонтальные связи по обмену информацией. В определенные часы узлы автоматически связываются друг с другом и обновляют свои банки данных и почты. Узлы также могут запрашивать информацию для своих пользователей у других узлов в произвольном режиме.

На базе каждого узла в регионах организуются точки. Каждая точка обменивается информацией со своим узлом (с узлом, к которому она прикреплена) в автоматическом режиме, а также может запрашивать интересующую ее информацию в произвольном режиме.

Запрос и получение информации точкой от «чужого» узла происходит опосредованно, т. е. через «свой» узел. Адрес точки формируется на основе адреса обслуживающего ее узла.

Например, точка, установленная в школе № 69, может иметь сетевой адрес: 600:201/3.69.

Наконец, каждый обладатель необходимой для подключения к сети техники может стать абонентом сети в режиме терминала. В этом случае абонент не имеет сетевого адреса и поэтому лишен преимуществ обмена информацией в автоматическом режиме. Пользователь получает доступ к информации, находящейся в узлах и точках, только по своей инициативе.



Таким образом, структура сети четко определена на уровне узлов и точек, которые имеют сетевые адреса, а между ними производится обмен информацией по фиксированным правилам и в определенное время. Абоненты, работающие в режиме терминалов, представляют собой свободных пользователей.

2. Аппаратно-программное обеспечение сети

Для работы в телекоммуникационной сети необходимо иметь персональный компьютер (желательно IBM-совместимый). Сетевой узел должен быть оборудован достаточно быстрым (PC/386, PC/AT) компьютером с винчестером не менее 40 Мбайт, позволяющим хранить большие объемы информации. Точка может иметь компьютер типа PC/XT/AT с винчестером 20/10 Мбайт. Для работы абонента в режиме терминала можно использовать практически любой компьютер, имеющий аппаратно-программную возможность подключения к сети.

В качестве информационных каналов возможно использование по крайней мере двух технически принципиально различных линий связи. Создание компьютерной телекоммуникационной сети с использованием обычных телефонных каналов (скорость передачи информации по таким каналам составляет от 300 до 2400 бод) предусматривает использование уже существующих технологий и аппаратных средств. Для ее реализации каждому абоненту данной сети достаточно приобрести и установить модем. Для более устойчивой работы сетевых узлов целесообразно использование модемов с коррекцией ошибок передачи (MNP — модемы).

Необходимое программное обеспечение для работы компьютера с модемом легкодоступно и распространяется на некоммерческой основе. Для реализации функций сетевого узла, точки и терминала целесообразно использовать коммуникационные программы различного уровня сложности. Соответственно различаются и требования к уровню квалификации абонентов: от системного оператора на узле до квалифицированного пользователя на терминале.

Таким образом, достоинством данного варианта является его широкое распространение во всем мире, наличие необходимых аппаратных и программных средств, сравнительно недорогая стоимость, что обеспечивает возможность быстрого развертывания телекоммуникационной сети. К недостаткам можно отнести сравнительно невысокое качество советских телефонных каналов и их загруженность в дневное время, что приводит иногда к сбоям и ошибкам при передаче информации.

Компьютерная телекоммуникационная сеть на основе использования спутниковой системы радиосвязи использует принципиально новые технологии, аппаратные и программные средства. Информационный радиоканал между географически удаленными узлами сети реализуется с помощью спутников. В региональных центрах (узлах) устанавливаются мощные антенны, комплексы приемо-передающей аппаратуры, а также аппаратура, согласующая систему спутниковой связи с компьютером.

Для связи центров (узлов) с абонентами (точками и терминалами) в каждом из них должны быть установлены

радиоантенны и приемо-передающая аппаратура. Функционирование сети обеспечивается соответствующим программным обеспечением.

Достоинством данного варианта является высокая скорость передачи информации и независимость от существующих систем связи.

В данном проекте предполагается использование и объединение обоих вышеописанных каналов связи на основе единого информационного содержания и организационной структуры сети.

Общей проблемой для обоих вариантов развертывания телекоммуникационной компьютерной сети является ее ориентация на IBM-совместимые компьютеры. Подавляющее большинство установленных в учреждениях народного образования компьютеров таковыми не являются и при помощи стандартных аппаратно-программных средств к сети подключены быть не могут. Следовательно, одной из задач данного проекта обязательно должна быть разработка таких средств.

3. Функционирование телекоммуникационной сети

Целью функционирования телекоммуникационной сети для учреждений народного образования является резкая интенсификация информационных обменов для все более широкого круга абонентов. В качестве абонентов могут выступать учителя, преподаватели, администраторы и учащиеся школ, лицеев, гимназий, ПТУ, техникумов, а также методисты институтов усовершенствования и повышения квалификации.

Тысячи учителей и сотни методистов ведут постоянный поиск новых форм и методов обучения по всем предметам общеобразовательной подготовки. Однако результаты их труда во многих случаях остаются не известными подавляющему большинству потенциальных потребителей, если используются традиционные способы распространения (курсы в ИУУ, предметные журналы и т. д.).

Телекоммуникационная сеть делает методические материалы доступными для любого абонента. Например, методические материалы, разработанные в институте усовершенствования учителей, заносятся в память узла сети, функционирующего в том же институте, и становятся доступными всем абонентам сети. Любой учитель, подключившись к телекоммуникационной сети, может запросить перечень всех материалов по интересующей его теме, и выбрав любой из них, получить его.

Для реализации таких информационных обменов необходимо проведение телеконференций по определенным темам, примерный перечень которых на начальном этапе может быть следующий.

1. Курс информатики.
2. Предметы естественно-математического цикла.
3. Предметы гуманитарного цикла.
4. Нормативные документы по новым типам (гимназии, лицеи и т. п.) учебных заведений.
5. Деловые предложения.
6. Дискуссии между учащимися.

Информация по каждой теме может и должна быть структурирована, например материалы первой темы могут быть разбиты на следующие области:

1.1. Методика преподавания информатики.
1.2. Программное обеспечение.
1.3. Программы, планирования.
1.4. Контрольные работы, экзамены.
1.5. Нормативные документы.

Организация телеконференции по определенной теме состоит в следующем. Каждый зарегистрированный в сети абонент (узел или точка, имеющие сетевые номера) ее «объявляют», т. е. заносят в каталог, отводят место на дисковом пространстве и т. д. Абоненты сети посылают свои материалы по данной теме на «ближайшую» точку, а та передает их на «свой» узел. Узлы обмениваются поступающей информацией в автоматическом режиме и, таким образом, на всех узлах накапливается индентичная информация по данной теме.

Координатор данной темы периодически упорядочивает материалы по данной теме, т. е. стирает второстепенные и устаревшие материалы и фактически формирует банк информации по данной проблеме. Любой незарегистрированный абонент может запросить информацию из этого банка информации.

Наряду с телеконференциями по по-

стоянным темам возможна организация свободных временных дискуссий по интересующим темам. Такое неформальное общение особенно привлекательно для учащихся, так как в этом процессе реализуется принцип свободного обмена мнениями, они учатся культуре диалога, спора, обмена мнениями.

Одной из главных целей создания телекоммуникационной сети является удовлетворение практических потребностей учреждений народного образования в общедоступном банке программных средств.

Банк должен иметь фонд программных средств, в правила функционирования которого должен быть заложен механизм, стимулирующий отдельных разработчиков (учителей и учащихся) вкладывать в него свои программы. Такой механизм может использовать систему приоритетов. Приоритет абонента устанавливается в зависимости от количества программных средств, которые он сам вложил в банк. Соответственно каждый абонент сети на основе своего приоритета может получить то или иное количество программных средств.

Кроме механизма приоритетов необходимо будет обеспечить доступ к программным средствам любому абоненту за плату. Такая плата (например, за месяц или за количество полученных программ) должна быть небольшой и доступной для любого учреждения народного образования.

Коммерческий доход телекоммуникационной сети может быть частично использован для выплаты гонораров авторам наиболее удачных программ, причем ценность программ будет оцениваться не экспертами (где часто велик субъективный элемент), а на основе потребительского спроса. Чем больше абонентов запросили данный программный продукт, тем больше будет авторский гонорар, что будет являться дополнительным стимулом авторам для вклада своих программных средств в банк.

Такой банк может иметь коммерческую информационно-справочную систему по программному обеспечению, в которой должна быть заложена информация о том, какая организация и за какую цену предлагает ту или иную программу, а также краткая аннотация к данной программе. Такая система может функционировать на коммерческой основе:

продавец платит за помещение информации о своих ПС в данную систему;

потенциальный покупатель платит за доступ к заложенной информации.

4. Технология деятельности по созданию сети

Создание компьютерной телекоммуникационной сети должно проходить поэтапно и ориентироваться на практические потребности и возможности системы народного образования. Первый этап должен состоять, по-видимому, в создании Центра компьютерной телекоммуникационной сети на базе Московского института повышения квалификации работников образования и нескольких региональных центров на базе ИУУ, которые в структуре сети выполняли бы функции узлов. Вокруг каждого из этих узлов постепенно формировались бы точки.

Потенциально войти в сеть и стать полноправным абонентом может любой владелец необходимого аппаратного и программного обеспечения. Однако практическая работа в сети требует определенных знаний по общим основам функционирования телекоммуникационных сетей, навыков работы с компьютером и с сетевым программным обеспечением. Таким образом, желательно, чтобы потенциальный абонент сети прошел первоначальное обучение.

В процессе обучения потенциальному абоненту сети должно передаваться необходимое программно-методическое обеспечение, оказываться консультационная и практическая помощь в приобретении необходимых аппаратных средств. Практическое включение абонента в сеть в качестве точки происходит путем присвоения ему сетевого адреса.

Практическая работа абонента в статусе точки дает ему необходимый практический опыт, а также «испытывает» серьезность его намерений по долговременной работе в сети. Действительно, работа в сети — это не только большое удобство, но и ответственность. Абонент сети должен строго следовать

правилам обмена информацией в данной сети (подключать компьютер к информационному каналу в определенное время, обрабатывать почту и т. д.).

Абонентам сети, желающим повысить свой статус и преобразовать точку в узел, планируется предоставить возможность пройти дополнительное обучение и приобрести квалификацию системного оператора сетевого узла. Новый статус позволит абоненту создать свой банк данных и способствовать возникновению новых абонентов сети, управляемых им точек.

Ясно, что создаваемая телекоммуникационная сеть является саморазвивающейся и самоорганизующейся системой, в которой сетевые узлы являются «точками» роста. Данную статью можно рассматривать как первоначальный толчок, инициирующий процесс саморазвития системы.

С. БУРЦЕВ, А. ЕФРЕМОВ, Д. ЕФРЕМОВ, А. ЗОРИЧ

Комплект учебных миров «КуМир»

Памяти А. П. Ершова

1. Введение

В статье рассказывается о новой версии пакета учебных программ КуМир, поддерживающего школьный курс информатики. В пакет КуМир входят редактор-компилятор языка Е (школьного алгоритмического языка) и четыре исполнителя. Команды языка могут быть дополнены командами любого из этих исполнителей. В последнем случае мы будем говорить о КуМире+Исполнитель, или просто о КуМире+.

Идеи, положенные в основу КуМира+, развивают идеи языка ЛОГО и исполнителей Робот и Чертежник, описанных в школьном учебнике информатики. С другой стороны, КуМир является расширением Е-практикума [1—3], написанного в 1985 г. Д. В. Варсанофьевым и А. Г. Дымченко и дополненного затем А. Г. Леоновым и М. Г. Эпиктетовым, и сохраняет важнейшие его черты: на полях алгоритма немедленно выдаются сообщения обо всех синтаксических ошибках, алгоритм можно немедленно выполнить и результаты выполнения сразу будут видны на экране. КуМир+ дает в руки ученику некоторый Мир, который можно исследовать и изменять в процессе выполнения алгоритмов. Этот Мир должен быть достаточно прост, чтобы ученик мог быстро понять его устройство, и достаточно богат, чтобы можно было придумать много задач, специфических именно для этого Мира. Итак, КуМир — это несколько таких Миров, простой современный язык программирования и удобная система программирования на нем. Более ранняя версия КуМира+ описана в [4].

В этой статье описан КуМир для «Ямахи». Имеются также версии КуМира для ПЭВМ «Корвет» и IBM PC. При этом версия для IBM PC основана на новой реализации школьного алгоритмического языка, разработанной М. Г. Эпиктетовым (см. [5]).

КуМир включает в себя четыре Мира — Мир Строителя, Мир Вездехода, Мир Двунога и Мир Редактора. Сначала коротко опишем эти Миры.

Строитель живет на разграфленной, ориентированной по сторонам света плоскости, в клетках которой могут располагаться кубики. Кубики на поле Строителя могут стоять друг на друге. Строитель может прыгать вниз, но забираться вверх он может только на ступеньку высотой в один кубик. Кроме того, Строитель умеет перемещать кубики — отсюда его название — и определять абсолютную высоту местности вокруг себя.

Вездеход движется по полю, на котором располагаются прямолинейные стены произвольной длины. Вездеход перемещается на этом поле в любом направлении, не задевая за стены. На Вездеходе установлен локатор, с помощью которого можно измерять расстояние до ближайшей стены в направлении движения Вездехода. В поле заданы координаты, и Вездеход умеет так-

же определять свое положение на плоскости и направление движения (отклонение от направления оси абсцисс).

Двуног — это существо с двумя ногами и тяжелой головой на длинной шее. Вместо туловища у него центральный шарнир, вокруг которого могут независимо вращаться ноги и шея. Живет Двуног в плоском (двумерном) Мире. В каждый момент времени Двуног какой-то своей конечностью касается земли. На ногах у Двунога есть присоски, которыми он может прикрепляться к земле (и даже к потолку). Если присоски отпущены, Двуног падает под действием силы тяжести.

Редактор умеет обрабатывать тексты, состоящие из строк одной и той же длины. По тексту перемещается светящийся прямоугольник — курсор, отмечающий текущее положение в тексте. Редактор умеет перемещать курсор и вносить изменения в текст: вставлять и удалять символы и строки, запоминать текущий символ и вспоминать запомненный символ.

В КуМире для этих Миров реализованы принципиально новые возможности по сравнению с Роботом, Чертежником или Черепашкой (см. [6]). Это — исполнители — Конструкторы Мира и возможность ручного управления Миром. Они, с одной стороны, позволяют прояснить для ученика процесс создания алгоритма, а с другой — дают широкое поле для независимой деятельности по крайней мере двух учеников в одном и том же Мире.

В дальнейшем мы подробно обсудим понятие Мира, возможности Конструкторов и ручного управления. Будут еще раз даны более подробные описания четырех Миров, входящих в КуМир, для каждого из них мы приведем систему команд исполнителя и Конструктора, примеры алгоритмов и несколько задач.

2. КуМир и изучение информатики

Приведем несколько соображений, которые, на наш взгляд, показывают важность КуМира.

2.1. КуМир — новая сфера деятельности

Ядро курса информатики — изучение алгоритмического языка, т. е. еще одной знаковой системы. При использовании наглядных Миров эта система выгодно отличается от знаковой системы математики своей конструктивностью и приближенностью к житейскому опыту учащегося. Число решенных на компьютере задач и упражнений для каждого Мира может измеряться десятками. Такая работа, безусловно, развивает творческие способности, дисциплинирует мышление.

Но эту работу нельзя назвать математической. Информатика похожа на математику, но не сводится к ней и имеет свою специфику. Многие способные, сообразительные школьники не преуспевают в математике (скорее всего, она для них слишком абстрактна), но начинают успешно и с большим удовольствием заниматься информатикой. Более того, в каждом классе выдвигаются новые лидеры, улучшающие свое положение в классе за счет успехов по информатике, успехов в освоении компьютера. Иногда решение задач на компьютере с привлечением наглядных Миров становится для школьников столь же увлекательным, как компьютерные игры. Компьютерная сфера становится заметной частью школьного фольклора, поводом для «профессионального» общения.

Работая в каждом Мире, можно поставить широкий спектр задач, начиная от элементарных упражнений на закрепление материала и кончая трудными творческими задачами олимпиадного уровня сложности. Это позволяет учителю варьировать глубину изучения материала от ученика к ученику, сохраняя единство деятельности для всего класса, избегая появления отстающих. Каждый достигает успеха на каждом занятии.

При решении задач для самых разных Миров математика часто выступает как удобное подручное средство, лишенное ореола трудности или абстрактности. Поэтому занятия информатикой могут способствовать привыканию, повышению интереса, лучшему освоению математики.

Работа с КуМиром заведомо доступна семиклассникам, вызывает у них большой интерес.

2.2. Чему может научить КуМир

Наши исполнители — это простейшие примеры систем, моделирующих те или иные области человеческой деятельности. Чтобы управление системой было доступно школьнику, приходится отказаться от какого бы то ни было правдоподобия: так, в существующих относительно правдоподобных моделях шагающих механизмов число узлов достигает нескольких десятков, причем приходится использовать шарниры разных типов (см., например, [7]). Управление такой объемной системой очень сложно. С Двуногом же могут справиться школьники самого разного уровня программистской подготовки. Начинать можно с простейших линейных алгоритмов по 4—6 команд, обучая Двунога каким-то элементарным действиям (шаг, сесть, встать ...). При этом программа создается в значительной степени подбором подходящих параметров в командах. На более высоком уровне, когда возникает потребность повторять какие-то действия (например, шаги), начинается использование оператора цикла, вспомогательного алгоритма и алгоритма с аргументом. Те же соображения относятся к другим исполнителям.

Принципиально другой уровень сложности представляют универсальные алгоритмы, т. е. алгоритмы, предназначенные для широкого классах возможных рельефов, или возможных текстов, или возможных лабиринтов. Здесь необходимы разветвленные алгоритмы, использующие обратную связь. При этом предшествующая работа по созданию линейных алгоритмов, как правило, не пропадает — приходится использовать простые линейные алгоритмы для элементарных действий вроде шага.

Отметим, что универсальные алгоритмы не только делятся на правильные и неправильные. Они могут существенно отличаться по качеству: например, один алгоритм движения Двунога по пересеченной местности годится только для какого-то элементарного рельефа, другой — практически для любого. Эту типичную для реальной жизни ситуацию, когда задача в общем виде неразрешима, но может быть решена в той или иной степени приближенно, редко удается продемонстрировать школьникам. Между тем именно при работе с такого рода задачами начинается творчество, возникает интерес и, что самое главное, появляется ощущение свободы и уверенности в своих силах.

2.3. Новые возможности КуМира

Примерами совместной деятельности являются задачи типа «шифровка — дешифровка», в которых один ученик шифрует текст, а другой его расшифровывает, один создает лабиринт на поле Робота, другой — алгоритм для прохождения этого лабиринта и т. д. В принципе возможны задачи типа «шифровка — перешифровка — дешифровка» и т. д. Подобные задачи вносят в работу элементы соревнования, ребятам становится интересно, они активнее работают и в результате лучше понимают материал. Для поддержки совместной работы в КуМире к каждому исполнителю добавлен Конструктор Мира, который общается с тем же Миром, но имеет значительно более широкие возможности для его изменения. Например, Двуног может лишь ощупывать рельеф и передвигаться по нему, а Конструктор — создавать новые ступени, туннели, мосты и т. д. В общем, когда один ученик составляет алгоритм для исполнителя, а другой — для Конструктора, они изменяют разные стороны одного Мира, причем конечный результат зависит от усилий обоих и «сфера влияния» одного недоступна другому.

Кроме того, наш опыт показывает, что, получив задание составить алгоритм, ученик чаще всего сразу пытается обратиться к более или менее мощным конструкциям алгоритмического языка. При этом, однако, он не всегда ясно представляет себе идею решения, не говоря уже о четком, но неформальном описании алгоритма. Поэтому представляется крайне важным дать в руки ученику инструмент, пользуясь которым он мог бы нащупать решение на идейном уровне. Таким инструментом является ручное управление исполнителем. С его помощью можно исследовать и изменять Мир не только в процессе выполнения алгоритма, но и в процессе непо-

средственной работы за компьютером, когда видны результаты каждого действия, и на их основе можно выбирать дальнейшую стратегию. Это в свою очередь позволяет сформировать представление о способе решения задачи, который предстоит записать средствами алгоритмического языка. Таким образом, оказывается возможным явно разделить две важные стадии процесса построения алгоритма.

3. Что такое Мир

Представим себе, что у нас есть некоторое устройство, которое может функционировать в определенной обстановке и умеет выполнять фиксированный набор команд. Это может быть как реальный механизм, так и модель такого механизма, изображаемая на экране ЭВМ. Набор команд такого устройства может включать в себя команды управления и команды, позволяющие определить текущее состояние обстановки, в которой действует устройство. Вообще говоря, способ, которым команды непосредственно передаются устройству, для нас несуществен: возможно (если устройство — это какой-то механизм), путем нажатия на кнопки, расположенные на нем или на пульте, соединенном проводами с устройством (как на детской игрушке типа лунохода), возможно, иным способом. Такое устройство мы будем называть исполнителем; команды, которые оно умеет выполнять,— системой команд исполнителя. Каждый исполнитель работает в определенной обстановке. КуМир+ в каждый момент «помнит» два состояния обстановки — начальное и текущее. Мир — это исполнитель, его система команд и обстановка, а также Конструктор и его система команд.

Если передавать команды исполнителю и получать от него информацию можно с помощью ЭВМ, то нам, вообще говоря, неважно, в каком именно виде существует наш Мир, «реален» он или нет: ведь «правила игры» уже определены. Наши Миры существуют на экране компьютера. При этом на верхней половине экрана изображаются обстановка исполнителя и он сам, а на нижней половине виден текст алгоритма. Все изменения Мира, происходящие при выполнении алгоритма, немедленно отображаются на экране. Такой способ изображения Мира присутствовал уже в реализации Робота и Чертежника.

Опишем подробнее, как именно можно работать с Мирами. Разумеется, можно составлять новые алгоритмы управления исполнителями и сохранять их на диске. Можно прочитать с диска уже готовый алгоритм и внести в него изменения. Это возможности изменения текста алгоритмов.

Несколько слов об управлении исполнителями. Так же как для Робота и Чертежника, существует возможность непрерывного и пошагового выполнения алгоритмов. Кроме того, в КуМире есть еще одна возможность — ручное управление исполнителем. В этом режиме можно выполнять все команды исполнителя с помощью нажатия на те или иные клавиши. Каждой команде соответствует одна клавиша, при нажатии на которую эта команда немедленно выполняется и результат виден на экране.

Для того чтобы исполнитель мог работать в какой-то обстановке, эту обстановку надо создать. Созданием обстановки может быть постановка новых стен в поле Вездехода, или создание нового рельефа для Двунога, или подготовка текста для обработки Редактором, или что-нибудь еще. Можно также двигать Вездеход в его поле, ставить Двунога в какую-нибудь позу, ставить Строителя на какой-то из кубиков. Всем этим, как уже упоминалось, занимается Конструктор Мира, для которого тоже предусмотрена возможность ручного управления. Ручное управление Конструктором будем иногда называть ручным изменением обстановки.

4. Конструкторы Миров

В описываемой версии КуМира для каждого из Миров кроме Мира Редактора есть два файла. В одном из них реализовано ручное и автоматическое управление исполнителем и ручное изменение обстановки, а в другом — ручное и автоматическое управление Конструктором. Такое разделение отчасти объясняется ограниченными возможностями

«Ямахи», а отчасти — желанием разделить «сферы влияния» учеников в задачах по совместной деятельности. Команды Конструктора Мира Редактора совпадают с командами самого Редактора, поэтому весь Мир Редактора реализован в одном файле.

Загрузка Конструкторов и работа с ними происходят по тем же правилам, что и исполнителями. Команды Конструкторов можно включать в алгоритмы КуМира+, есть возможность сохранять результаты их работы на диске и использовать полученную обстановку при работе с исполнителем. Другими словами, Конструкторы являются самостоятельными исполнителями. Название «Конструктор» подчеркивает прежде всего то, что он не образует собственный Мир, а дает возможность составлять алгоритмы изменения обстановки соответствующего исполнителя.

Конструкторы обеспечивают качественно новую возможность обучения информатике — совместную, но независимую работу двух или нескольких учеников. Один из них может писать алгоритм построения состояния обстановки, удовлетворяющего каким-то условиям, а другой — алгоритм управления исполнителем, в котором это состояние принимается за начальное. Например, один ученик пишет алгоритм, составляющий сложный рельеф для Двунога, а другой — алгоритм, при выполнении которого Двуног шагает по этому рельефу. Или один ученик создает алгоритм, который строит лабиринт из стен в поле Вездехода, а другой проводит Вездеход по этому лабиринту. Или один составляет алгоритм шифровки текста при помощи Редактора, а другой — алгоритм дешифровки. Класс может быть разделен на две группы, одна из которых пишет алгоритмы для исполнителя, а другая — для Конструктора. Чей Двуног быстрее пройдет по заданному рельефу? Какой рельеф окажется самым труднопроходимым? Чей Вездеход заблудится в простом лабиринте, а чей сможет выйти из самого сложного? Кому удастся расшифровать зашифрованный текст без ошибок? Такой стиль обучения безусловно интересен для школьников, активизирует их творческие способности.

5. Ручное управление

В этом небольшом разделе мы суммируем причины, по которым введено ручное управление. Во-первых, когда знакомство с Миром только начинается, очень полезно попробовать вручную, как выполняются те или иные команды. Конечно, можно посмотреть работу исполнителя с помощью простых алгоритмов из одной-двух команд, но это довольно долго и не так наглядно, как ручное управление. Новички к тому же могут избежать дополнительных трудностей, связанных с одновременным освоением конструкций языка и команд исполнителя.

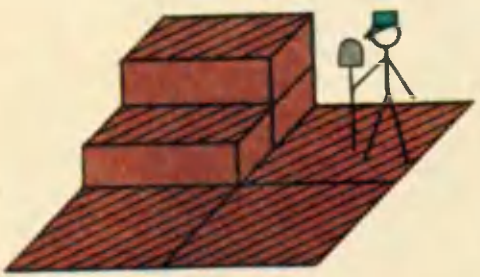
Во-вторых, как уже обсуждалось выше, ручное управление дает возможность экспериментировать с исполнителем в процессе решения той или иной задачи. Это помогает найти идею построения алгоритма. Наконец, ручное управление позволяет сочетать непосредственное и программируемое управление исполнителем. Это дает ученику возможность почувствовать на собственном опыте, как можно повысить эффективность работы при помощи ЭВМ, если переложить на нее техническую часть работы. Предположим, что ученик экспериментирует с исполнителем, обдумывая какой-то сложный алгоритм, и уяснил себе, как выполнять тот или иной часто встречающийся фрагмент (например, как Строителю перекладывать столбик, стоящий перед ним, за себя). Тогда он может написать алгоритм, выполняющий этот фрагмент, и в нужные моменты просто выполнять его, сосредоточив все усилия на более содержательной части задачи.

Конструктором тоже можно управлять вручную: во-первых, это нужно при создании обстановок для отладки алгоритмов для исполнителей. Кроме того, справедливы те же соображения, что и для ручного управления исполнителем. В частности, при создании сложной обстановки какие-то части можно строить вручную, а какие-то — с помощью выполнения алгоритма. Например, можно составить алгоритм «сделать ступеньку» для Конструктора Двунога и выполнять его каждый раз, когда нужно сделать ступеньку в рельефе, а в ос-

тальном строить рельеф вручную. Или составить алгоритм «построить траншею» для Конструктора Строителя. Высота ступеньки, глубина траншеи или ее длина могут быть аргументами такого алгоритма.

6. Мир Строителя

Представим себе прямоугольную комнату, сориентированную по сторонам света: одна стена северная, другая — южная, третья — западная, четвертая — восточная. Пол этой комнаты расчерчен на клетки, на них стоят столбики из одинаковых кубиков. Высота столбиков не больше некоторой максимальной. Их поверхность вместе с поверхностью пола (там, где нет кубиков) образует некоторый рельеф с дискретными уровнями высоты. Строитель может ходить по этому рельефу, при этом за один шаг он переходит на один из соседних столбиков. Он также может перекладывать кубики по определенным правилам. Кроме того, Строитель может узнать высоту столбика, на котором он стоит, и соседних столбиков, а также отличить ситуацию, когда он стоит около стены комнаты. Строитель всегда смотрит в направлении одной из сторон света. Он готов сделать шаг в этом направлении. Будем называть это направление текущим, столбик, на котором стоит Строитель в данный момент, также назовем текущим. Наконец, Строитель имеет компас, так что при необходимости он может сориентироваться. Обстановка Строителя определяется высотами столбиков, а его состояние — текущим столбиком и текущим направлением. Кусочек комнаты со Строителем изображен на рис. 1.



Как мы видим, Строитель способен изменять обстановку. Однако, несмотря на все изменения, Строитель помнит начальное состояние обстановки. Приведем список команд Строителя.

строитель	на__юг	цел	высота
шаг__вперед	на__запад	цел	высота__спереди
налево	на__восток	цел	высота__сзади
направо	положить	цел	высота__справа
кругом	взять	цел	высота__слева
на__север		лог	можно__идти

Команда «строитель» возвращает обстановку в начальное состояние. По командам: «шаг__вперед», «налево», «направо», «кругом» — Строитель двигается и выполняет повороты.

В процессе работы Строитель забывает, в какую сторону он смотрит, однако он всегда может посмотреть на компас и повернуться в любую сторону. Для этого предназначены команды: «на__север», «на__юг», «на__запад», «на__восток».

По команде «положить» Строитель перекладывает верхний кубик с текущего столбика вперед, а по команде «взять» — верхний кубик со столбика перед собой на текущий столбик.

Остальные команды осуществляют обратную связь. Команда «высота» возвращает высоту столбика под Строителем, следующие четыре — высоты столбиков вокруг него; последняя команда проверяет, может ли Строитель переместиться на кубик перед ним.

Перейдем к Конструктору Мира Строителя. Вот его команды:

начать	к__юг	налево	
поставить	к__зап	цел	выс
снять	с__сев	цел	выс__с
колонка (цел а)	с__вос	цел	выс__в
к__сев	с__юг	цел	выс__ю
к__вос	с__зап	цел	выс__з

Команда «начать» восстанавливает начальное состояние Мира Строителя. Текущая клетка выделена курсором. Команды: «поставить/снять» — поставить/снять кубик в текущей клетке; «колонка» — поставить столбик в текущей клетке; «к__...» («с__...») — сдвинуть курсор (Строителя) на одну клетку в указанном направлении. Команды обратной связи позволяют определить высоту столбика в текущей клетке и в четырех соседних.

Продолжение следует.

Д. БОГОЛЮБОВ
МИЭМ, Москва

Использование Пролога в гуманитарных предметах

Основное внимание в «традиционных» курсах информатики уделяется алгоритмизации, основам процедурного программирования и решению вычислительных задач, поэтому они мало пригодны для учащихся гуманитарных классов и гуманитарных специальностей в вузах.

Это объясняется отсутствием учебного языка программирования, с одной стороны, доступного для учащихся с гуманитарным складом ума, а с другой стороны, являющегося удобным инструментом для решения нечисловых задач.

Для гуманитарных классов и специальностей вузов разрабатывается курс информатики, в основу которого положено изучение логики и решение задач на языке Пролог. Цель курса — овладение учащимися компьютерной грамотностью (читать, писать, рисовать, считать, музицировать, искать информацию с помощью ЭВМ), формирование у учащихся логического мышления (а не алгоритмического) и применение полученных знаний, умений и навыков при изучении гуманитарных предметов. Первую часть занимает введение в информатику и компьютерную грамотность, в которую входит знакомство с ПЭВМ, с видами информации, работа с текстовым, графическим, музыкальным редакторами, электронными таблицами и базами данных. Для ПЭВМ «Русич» и IBM/PC специально разработан интегрированный учебный пакет «Компьютерная грамотность» с унифицированным дружественным интерфейсом. Во второй части курса учащиеся изучают элементы содержательной и формальной логики параллельно с логическим программированием на языке Пролог. В качестве компьютерной поддержки используется разработанный автором «Электронный задачник по логическому программированию на языке Пролог-Д». Третью часть составляет знакомство с использованием ЭВМ в обществе и практикум по решению задач из гуманитарной области. Учащиеся решают лингвистические задачи, строят базы знаний и информационно-логические модели, знакомятся с простыми экспертными и гипертекстовыми системами.

Рассмотрим несколько примеров использования языка Пролог в гуманитарных дисциплинах. Для записи программ будем использовать русскоязычную версию языка — Пролог-Д, подробно описанную в предыдущих номерах журнала.

Языки, грамматики, словари, или применение языка Пролог в языкознании

Понятие языка является одним из фундаментальных в курсах информатики. При изучении этого понятия применяется логический подход, рассматривающий язык как знаковую систему. В науке о знаках — семиотике выделяют три раздела: синтаксис, семантику и прагматику. Обычно в курсах информатики и программирования изучаются так называемые формальные языки, частным случаем которых являются языки программирования. Математическая теория таких языков хорошо разработана. Для описания синтаксиса формальных языков чаще всего используются нормальные формы Бэкуса — Наура, синтаксические диаграммы и формальные (порождающие) грамматики.

Опыт преподавания курсов информатики показал, что первоначальное знакомство с примерами описаний синтаксиса простых языков и грамматическим разбором значительно облегчает усвоение языка Пролог. Интересной особенностью грамматических правил является наличие в них рекурсий, т. е. использование для описания некоторых конструкций самих описываемых конструкций.

Пример грамматики на языке нормальных форм Бэкуса.

(1) <целое> ::= <цифра>|<целое> <цифра>
(2) <цифра> ::=∅|1|2|3|4|5|6|7|8|9

В этом примере в правиле (1) конструкция <целое> определяется через саму себя. Число 139 можно вывести следующим образом: <целое>→<целое> <цифра>→<целое> <цифра> <цифра>→<цифра> <цифра> <цифра>→1 <цифра> <цифра>→ 13 <цифра>→139.

Приобретенные навыки работы с «определениями», в том числе и рекурсивными, переносятся в дальнейшем в логическое программирование.

При изучении логического программирования на языке Пролог в качестве примеров и для самостоятельного решения могут предлагаться задачи на словообразование, склонение, спряжение, синтез синтаксически правильных предложений, разбор (синтаксический анализ) языковых конструкций, перевод слов и фраз, кодирование текстов, а также различные языковые игры.

В качестве примера рассмотрим задачу перевода и ее решение на языке Пролог. Возьмем знаменитую фразу профессора-лингвиста Л. В. Щербы: *«Глокая куздра штеко будланула бокра и кудрячит бокренка»*, которая является иллюстрацией того факта, что правильно составленная фраза по законам грамматики из бессмысленных слов уже несет некоторую семантическую нагрузку и может пониматься хотя и широко, но все же ограниченно. Дадим «перевод» (интерпретацию) слов с помощью словаря.

```
Словарь:
словарь (глокая, злая);
словарь (куздра, тигрица);
словарь (штеко, сильно);
словарь (будланула, поранила);
словарь (бокра, буйвола);
словарь (кудрячит, треплет);
словарь (бокренка, буйволенка);
словарь («и», «и»);
```

Замечание. В последнем факте союз «и» заключили в кавычки, так как единичный буквенный символ воспринимается Прологом-Д как переменная.

С помощью рекурсивного правила организуем выделение слов из фразы, представленной списком, с выводом их перевода.

```
перевод ([])—ПС, ВЫВОД («КОНЕЦ ПЕРЕВОДА»);
перевод ([x:y]) ← словарь (x, z),
ВЫВОД (z), ВЫВОД (« »), перевод (y);
```

Запрос к полученной базе Пролога можно задать следующим образом:

```
?перевод ([глокая, куздра, штеко, будланула, бокра, «и», кудрячит, бокренка]);
```

При запуске полученной программы на экран будет выведен следующий результат:

```
злая тигрица сильно поранила буйвола
и треплет буйволенка
КОНЕЦ ПЕРЕВОДА
```

В данном примере учащийся может расширить словарь и составить новые фразы для перевода. В программе можно сделать удобный ввод предложений в привычном виде, а не в виде списка, воспользовавшись встроенными предикатами Пролога-Д для работы со строками.

Рассмотрим задачу синтаксического анализа и синтеза предложений на языке Пролог. Возьмем предложения «Глокая куздра будланула бокра» и «Голодная тигрица загрызла буйвола». Напишем грамматику для фраз такого типа.

<предложение> ::=<группа_подлежащего> <группа_сказуемого>
<группа_подлежащего> ::=<определение> <подлежащее>
<группа_сказуемого> ::= <сказуемое> <дополнение>

Запишем грамматику и словарь в виде правил и фактов Пролога (символ «%» — признак комментария).

```
% I — входной список слов
% O, R — выходной список слов
% структура предиката «объект»: объект
(Вход, Выход, Название_объекта)
объект (I, O, предложение)←
    объект (I, R, группа_подлежащего),
    объект (R, O, группа_сказуемого);
объект (I, O, группа_подлежащего)←
    объект (I, R, определение),
    объект (R, O, подлежащее);
объект (I, O, группа_сказуемого)←
    объект (I, R, сказуемое),
    объект (R, O, дополнение);
```

```
% словарь (Слово, Член_предложения)
словарь (глокая, определение);
словарь (голодная, определение);
словарь (куздра, подлежащее);
словарь (тигрица, подлежащее);
словарь (будланула, сказуемое);
словарь (загрызла, сказуемое);
словарь (бокра, дополнение);

словарь (буйвола, дополнение);
% правило связи предиката «объект»
с предикатом «словарь»
% [S:R] — входной список, R — остаток списка, N — член_предложения
объект ([S:R], R, N)←— словарь (S, N);
```

Проверим, является ли предложением фраза: «Глокая куздра загрызла буйвола».

```
?объект ([глокая, куздра, загрызла, буйвола], L, предложение);
```

Результат:

L=[], что является признаком успеха.

Проанализируем группу слов «глокая куздра».

```
?объект ([глокая, куздра], Остаток', Объект');
```

Результат:

```
Остаток=[]
Объект=группа_подлежащего
Остаток=[куздра]
Объект=определение
```

Сгенерируем всевозможные предложения.

```
?объект (X, [], предложение);
```

Результат:

```
X=[глокая, куздра, будланула, бокра]
X=[глокая, куздра, будланула, буйвола]
...
X=[голодная, тигрица, загрызла, буйвола]
```

В приведенном примере не сложно расширить грамматику и словарь, усовершенствовать программу, что обычно выполняется в качестве самостоятельных заданий. С вопросами анализа и синтеза текстов на естественных языках можно познакомиться в книге [9] и там же почерпнуть материал для других задач.

Базы данных и базы знаний

Современные информационные технологии базируются на понятии баз данных (БД), которые используются как самостоятельно, так и в составе различных информационных систем. Для организации и ведения баз данных были созданы специализированные программные средства — системы управления базами данных (СУБД). Результатом работ в области искусственного интеллекта по представлению и обработке знаний было появление в 80-е гг. нового класса систем — баз знаний и специализированных инструментальных средств, обеспечивающих создание, ведение и применение баз знаний,— систем управления базами знаний (СУБЗ).

Язык Пролог позволяет создавать как базы данных, так и базы знаний. Базу данных Пролога можно рассматривать как реляционную БД. На этом языке не представляет труда реализовать различные модели представления знаний: семантические сети, системы фреймов, продукционные системы, логические модели.

В качестве примера использования языка Пролог рассмотрим фрагменты учебной базы знаний по русской истории «Великие князья, цари и императоры».

```
правил («Елизавета Петровна», дочь, «Петр I», 1741, 1761);
правил («Петр III», внук, «Петр I», 1761, 1762);
правил («Екатерина II», жена, «Петр III», 1762, 1796);
правил («Павел I», сын, «Петр III», 1796, 1801);
правил («Александр I», сын, «Павел I», 1801, 1825);
правил («Николай I», сын, «Павел I», 1825, 1855);
правил («Александр II», сын, «Николай I», 1855, 1881);
правил («Александр III», сын, «Александр II», 1881, 1894);
правил («Николай II», сын, «Александр III», 1894, 1917);
прозвали («Екатерина II», Великая);
прозвали («Александр I», Благословенный);
прозвали («Александр II», Освободитель);
прозвали («Александр III», Миротворец);
императрица (X)←— правил (X, дочь, _, _, _);
императрица (X)←— правил (X, жена, _, _, _);
император (X) ←— правил (X, сын, _, _, _);
император (X)←— правил (X, внук, _, _, _);
годы_правления (X) ← правил (X, _, _, N, K), ВЫВОД (X, «правил(а) с», N, по, K);
сколько_правил (X) ← правил (X, _, _, N, K), СЛОЖЕНИЕ (S, N, K), ВЫВОД (X, «правил(а)», S, лет);
кто_правил (G) ← правил (X, _, _, N, K), НЕ (МЕНЬШЕ (G, N)), НЕ
```

(БОЛЬШЕ (G, K)), ВЫВОД («В», G, году, «правил(а)», X);

К этой базе знаний можно задавать различные запросы, научить ее правильно выбирать окончания мужского или женского рода у глагола «правил(а)» и т. д.

Учебные экспертные и гипертекстовые системы

Бурное развитие теории и практики искусственного интеллекта привело к появлению нового класса прикладных программных систем, основанных на знаниях,— экспертных систем (ЭС) и новой области информатики — инженерии знаний. С практической точки зрения ЭС — это программная система, решающая задачи в конкретной предметной области, моделируя рассуждения человека — эксперта. Обобщенная экспертная система включает в себя базу знаний, содержащую факты и правила конкретной предметной области, механизм вывода (решатель), блок приобретения знаний и блок объяснения. ЭС работает в двух режимах: приобретения знаний и решения задач. В режиме приобретения знаний эксперт общается с системой через посредничество инженера знаний. Эксперт, используя механизм приобретения знаний, загружает в систему знания, которые позволяют пользователю в режиме решения задач самостоятельно (без эксперта) находить решения и получать от системы объяснения, как они найдены (или почему не найдены).

ЭС получили очень широкое распространение в технике, естествознании, промышленности, экономике, медицине, военном деле. Началась разработка экспертных обучающих систем (ЭОС), реализующих функцию обучения на основе знаний экспертов (предметника, методиста и психолога). В области гуманитарных наук разработка ЭС сдерживается сложностью приобретения и формализации знаний. Наибольшие успехи здесь достигнуты в археологии, искусствоведении, психологии, юриспруденции. Для облегчения приобретения знаний в таких плохо формализуемых гуманитарных дисциплинах, как история, литература, философия, предлагается использовать гипертекстовую технологию [15].

Неструктурированная информация о предметной области (текстовая, графическая и т. д.) с помощью гипертекстовой оболочки превращается в семантическую сеть. Линейный текст становится гипертекстом, который может читаться не только последовательно «вертикально», но и с мгновенным переключением на сложные понятия «горизонтально». С одной и той же гипертекстовой моделью могут работать несколько экспертов-предметников и путем многократной проработки материала выделить основные понятия и разнообразные связи между ними. Полученный гипертекст может использоваться самостоятельно в виде учебной системы, а может служить прототипом для дальнейшей разработки учебной экспертной системы.

В качестве инструментальных средств построения экспертных систем используют «пустые» ЭС, оболочки ЭС, а также языки программирования и инженерии знаний. Последние годы широкое применение получил язык Пролог, на котором могут создаваться непосредственно и сами экспертные системы, и инструментальные средства создания ЭС. В школьном и вузовских курсах информатики на языке Пролог удобно писать учебные экспертные системы, т. е. упрощенные ЭС, частично или полностью разрабатываемые самими учащимися, или системы, служащие «интеллектуальными» учебными пособиями по какой-либо узкой предметной области. В качестве примера такой системы можно привести учебную экспертную систему по качественному стихосложению, разработанную на языке Пролог школьниками и студентами в Центре информатики МИЭМ. По введенному стихотворению со специально отмеченными ударными слогами система определяет тип стиха: силлабический, силлабо-тонический или тонический, а для силлабо-тонического стиха указывает, каким размером написано данное стихотворение (ямб, хорей, дактиль, амфибрахий, анапест). Система имеет механизм объяснения,

позволяющий учащемуся получить обоснование найденному решению, которое часто бывает далеко не очевидным даже преподавателям литературы.

С введением в учебный процесс IBM-совместимой техники экспертные системы будут получать все большее распространение.

О. ПАРСАДАНОВ

Роль задач и упражнений в курсе информатики

Подбор задач в любом учебном курсе неразрывно связан с его общими целями и производится исходя из его идейно-методической концепции. Говоря о курсе ОИВТ в школе, трудно признать наличие в настоящее время такого единого идейного подхода, и это обусловлено многими причинами, среди которых:

существенно разный уровень подготовки учителей (явление, беспрецедентное в сравнении с другими предметами);

разнотипность используемой техники;

различия в используемых учебниках.

Можно спорить, что должно являться предметом обучения в курсе ОИВТ, обсуждая разные варианты программ и учебных планов. Однако, пока, целью является формирование неких навыков алгоритмического мышления и обучение основам программирования. Приведем перечень тем, которые изучаются в большинстве школ:

1. Основы построения алгоритмов.
1.1. Линейные алгоритмы.
1.2. Разветвляющиеся алгоритмы.
1.3. Циклические алгоритмы.
2. Графические возможности ЭВМ.
3. Обработка массивов данных.
3.1. Одномерные массивы.
3.2. Двумерные массивы.
4. Практическое использование ЭВМ.
4.1. Экранные редакторы.
4.2. Системы управления базами данных.
4.3. Табличные процессоры.

Многим учителям становится все более ясно, что этот набор достаточно далек от современных требований.

С другой стороны, понимая ограниченность существующего подхода, реализовать обучение с упором на прикладное использование ЭВМ сложно из-за низкого уровня техники, «слабости» доступного программного обеспечения и отсутствия четкого понимания учителем целей и перспектив курса. Положение усугубляется тем, что настоятельным требованием времени стало снижение возраста обучаемых информатике, но ученики IV—VII классов не имеют нужной математической подготовки.

Прежде чем говорить о методах подбора задач при обучении курсу ОИВТ, надо сначала разобраться, чему и зачем учить. Этого нельзя сделать, если хотя бы бегло не проанализировать тенденции развития и использования современных средств вычислительной техники, что в принципе должно являться темой отдельной статьи. Очень кратко эти тенденции можно сформулировать в 5 пунктах:

1. Количество ПЭВМ в мире оценивается в 100—150 млн. штук, из них около 80 % IBM-совместимых. Важно отметить, что ЭВМ типа IBM PC/XT, чья удовлетворительная копия только недавно создана отечественной промышленностью и о которой в школах пока только мечтают, устарела и в мире практически уже не выпускается.

2. Технологии объектно и проблемно ориентированных подходов уже позволили создать новые способы организации интерфейса «Человек — ЭВМ», которые поддерживаются программно:

операционными системами типа OS/2 или UNIX V.5 и языками высо-

кого уровня типа C++, Smalktalk, Obenon и пр., а также аппаратно: процессорами I80386, I80486 и RISK-процессорами.

3. К 1993 г. ожидается создание процессора I80586 со степенью интеграции в 4 раза большей, чем у I80486, а к 1998 г. появится I80786 с быстродействием более 2 млрд. оп/сек. Вместе с тем уже сейчас имеют все большее распространение специальные DSP-процессоры (Digital Sound Processing — цифровая обработка звука) и другие специализированные аппаратные средства.

4. Такой рост вычислительных мощностей, позволяющий создать ПЭВМ, превосходящие по возможностям самые мощные из современных мини-ЭВМ, а также качественное изменение роли сетей ЭВМ безусловно приведет, с одной стороны, к появлению в конце 90-х г. г. принципиально нового интерфейса взаимодействия человека с ЭВМ, основанного на принципах цифровой видеоинтерактивной технологии (DVIT), распознающей речь, почерк, «оживляющей» графическое изображение и т. п., а с другой стороны — к появлению новых принципов архитектуры самой ПЭВМ. (Упомянем уже реализованные новые архитектурные подходы: MCA фирмы IBM или EISA фирмы COMPAQ.)

5. Кроме того, все сильнее растет влияние мощно развивающейся технологии автоматизации разработок программного обеспечения (Computer Aided Software Engineering), позволяющей создавать прикладные системы, не углубляясь в особенности архитектуры ЭВМ, реализаций языка программирования и протоколов сетевого обмена.

Из этого можно сделать следующие выводы:

1. Происходит бурная эволюция идеологии и принципов построения интерфейса «человек — компьютер».

2. Следует ожидать уменьшения спроса на «чистых программистов».

3. Появляются новые требования к квалификации работников, для которых становится главным:

осмысление принципов использования ЭВМ в своей деятельности;

понимание способов построения диалога с ЭВМ;

осмысление границ применения ЭВМ.

Рассматривая с этих позиций ситуацию с преподаванием ОИВТ в наших школах, можно более четко сказать, чего мы достигли, что отсутствует и каковы объективные проблемы.

Имеется:

5-летнее существование курса как «вещи в себе»;

излишне разнообразный «парк» школьных ЭВМ;

формирование групп, способных в принципе вести разработки программно-методического обеспечения;

«математизация» курса;

резкая поляризация учителей по уровню владения ЭВМ;

понимание факта, что другой техники и других учителей у нас нет и не скоро будут.

Отсутствует:

общая концепция курса;

осмысление учителями целей и задач курса;

техника и программные средства, соответствующие мировым стандартам;

идеология компьютеризированного обучения, т. е. использование новой техники для обучения другим предметам.

Предстоит:

снижать возрастную границу обучаемых основам информатики;

учителям, не слишком глубоко знакомым с ЭВМ, вести курс на морально устаревшей технике;

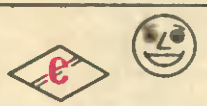
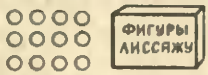
развивать содержание курса, учитывая всевозрастающие требования к уровню подготовки учащихся.

Таким образом, существующие учебные планы и имеющийся в школах парк ПЭВМ вряд ли способны в полной мере решить задачи подготовки школьников с учетом требований XXI в. Игнорировать сложившееся положение невозможно. Трудно ожидать также появления в ближайшее время какой-либо четкой официальной концепции обучения по предмету ИВТ.

Ниже рассматривается один из возможных вариантов снятия этих противоречий путем более внимательного

подбора задач и практических заданий в рамках традиционно сложившихся и утвержденных учебных планов.

Задачи и упражнения (рис.) должны служить не только и не столько средством закрепления определенных навыков и знаний, полученных на теоретических занятиях, но выполнять собственную роль как средства развития мышления ученика.

ТЕМА	ЗАДАЧА	ЗАДАЧА
ЛИНЕЙНОЕ ПРОГРАММИРОВАНИЕ	Y=A+B	
ВЕТВЛЕНИЕ	$Ax^2+Bx+C=0$ $Y=\begin{cases} Ax+B, x>B \\ Ax-B, x<B \end{cases}$	
ЦИКЛЫ	Y=x! Y=f(x)	 ПОСТРОЕНИЕ МЕНЮ
МАССИВЫ ДАННЫХ	$S=\sum_{i=1}^{n} A_n$ MAX, MIN элемент	ДВУХМЕРНЫЕ ЛИТЕРНЫЕ ТАБЛИЦЫ (ЖУРНАЛ, ВЕДОМОСТЬ) ЛОГИКА ПОСТРОЕНИЯ ЗАПРОСОВ ПРИ ВЫБОРЕ ДАННЫХ
СУБД	ДЕМО-ВЕРСИИ, ИГРЫ	ИДЕОЛОГИЯ ЯЗЫКОВ ЗАПРОСОВ ДАННЫХ

В качестве примеров можно рассмотреть некоторые темы типового учебного плана. Они могут использоваться независимо от типа техники. В них сохранена традиционная ориентация на изучение программирования. Выбор примеров обусловлен их простотой и ориентацией на большинство учителей.

1. *Линейные алгоритмы.* Обычно в этом разделе принято решать алгебраические задачи типа Y=A+B и т. п. Однако этому должно предшествовать довольно длительное изучение операторов ввода, вывода и присваивания, сопряженное с неизбежными трудностями непонимания учениками всего, что связано с новой для них логикой работы на ЭВМ и освоением новых синтаксических понятий языка программирования.

Вместо математических примеров можно было бы использовать задачи по рисованию интересных картинок, а рассмотрение собственно синтаксических конструкций оставить на самостоятельное изучение по учебникам, плакатам и стендам. При этом, как только два-три ученика в классе научатся использовать операторы графики, остальные школьники очень быстро перенимают эти умения у сильных учеников.

2. *Разветвляющиеся алгоритмы.* Обычно тут принято решать пресловутое «квадратное» уравнение и различные алгебраические неравенства. Понятно, что преобладание математических задач не способствует популярности предмета.

Вместо этого можно порекомендовать продолжать работы с графикой, рисуя концентрические окружности, расходящиеся лучи и т. п. Особое место могут занимать в этом разделе упражнения с получением на экране бегущих букв или слов методом закрашивания старого изображения цветом фона экрана.

Очень важно реализовать эти примеры именно в разделе «Ветвление», а не в разделе «Циклы», как это может показаться на первый взгляд. Вообще данный раздел удобно использовать как пропедевтику следующего раздела «Циклы».

Отдельное место занимают первые попытки реализации простейшего диалога с ЭВМ. Примитивная программа типа:

```
...
50 CLS
60 INPUT «Будете еще смотреть картинку (Д/Н)»; N$
70 IFN$=«Д» THEN 10 ELSE IF N$=«Н» THEN 90
80 GOTO 50
90 END
```

производит сильное впечатление на ученика, давая ему возможность ощутить и у себя способности управлять компьютером.

3. *Циклические алгоритмы.* Стандартно в этом разделе решаются задачи построения графиков тригонометрических функций, подсчета факториала числа и т. п. Фактически к этому моменту довольно большая часть учени-

ков по сути «выпадает» из учебного процесса, с одной стороны, из-за слабой математической подготовки, а с другой стороны — из-за непонимания принципов построения и выполнения программ на ЭВМ.

В этом разделе можно все еще продолжать упражнения с графикой, например, рисуя ряды окружностей или других картинок. При этом, можно сделать первый экскурс к теме «Подпрограммы», получая ряды одинаковых картинок как в цикле, так и используя подпрограммы. Для любителей алгебры можно предложить получение на экране фигур Лиссажу, которые выглядят гораздо привлекательнее обычной синусоиды.

Однако главным в этой теме является переход к построению простейших программ, генерирующих на экране меню с проходом по нему курсора. Как правило, для таких «мелочей» у учителя не хватает времени, а между тем создание меню требует написания совсем небольшой программы.

4. *Массивы данных*. Эта тема представлена задачами поиска суммы элементов массива, количества элементов больше нуля, максимального (минимального) элемента массива, нуля и т. д. Трудно спорить с тем, что они не вызывают у ученика ничего, кроме скуки.

Один из возможных вариантов изучения данной темы — работа с двумерными литерными таблицами и решение задач на обработку данных типа ведомостей, журналов, каталогов и т. п. Здесь уместно обсудить вообще способы хранения и представления информации в обычной жизни, делая акцент на таблицу как одно из основных средств записи информации.

Главным в этой теме, видимо, должно стать осмысление логики построения запросов при выборе данных и умение такие запросы реализовывать, готовясь к изучению следующей темы.

5. Системы управления базами данных. Обычно этот раздел посвящается работе с имеющимися СУБД и заполнен изучением описания пользователя вместе с тренировкой в нажатии соответствующих клавиш. Конечно, эта работа полезна, но фактически ученик только учится работать в среде данной СУБД, которая может очень сильно отличаться от любой другой такой же программы.

Существенно важным в этом разделе может и должно стать осмысление принципов построения СУБД вообще и идеологии языков запросов данных в частности. Конечно, не следует считать, что в школе требуется обязательное изучение таких вещей, как SQL (один из перспективных языков запросов данных), но подготовить школьника к работе с подобными средствами в будущем просто необходимо.

Таким образом, уделяя пристальное внимание выбору практических и контрольных заданий, работая на прежней технической базе, но по-новому осмыслив цели и задачи всего курса, учитель мог бы работать более эффективно. При этом основные требования к уровню знаний ученика можно сформулировать следующим образом.

1. Понимание принципов построения интерфейса «Человек — ЭВМ», а также «ЭВМ — ЭВМ»:

понимание сущностных различий естественных и формальных языков;

осмысление способов представления информации;

умение строить простой диалог с ЭВМ.

2. Представление о сущностных принципах функционирования ЭВМ:

отдельная ЭВМ;

объединенные в сеть ЭВМ.

3. Осмысление концепции использования ЭВМ:

навыки постановки задач для решения на ЭВМ;

понимание логики построения запросов к базам данных и умение реализовать эти запросы.

4. Понимание границ использования ЭВМ:

ограниченность возможностей ЭВМ;

области наиболее целесообразного использования ЭВМ.

Остается лишь добавить, что, внимательно осмыслив пункты 1 и 4, любознательный читатель сможет обнаружить, казалось бы, с трудом видную

связь информатики с предметами гуманитарного цикла. Речь идет о возможности обсуждения в рамках курса ОИВТ способов общения между людьми, проблем языков и моделей человеческого поведения. И именно такой поворот темы поможет лучше понять роль ОИВТ в общей системе учебно-воспитательного процесса в школе.

ЧТО МОЖЕТ ЭВМ?

Дорожный компаньон

Именно так назвали свою новую разработку на фирме DEK Sales Inc. Это небольшое устройство размером 54×86×4,3 мм можно было бы назвать калькулятором, но даже краткий обзор того, что может делать эта карманная машина, заставляет именовать его не иначе как специализированным компьютером.

На что же специализирован Travel Mate? На работу вашим личным бухгалтером. Все дорожные затраты разбиты на двадцать сфер: покупка билетов, еды, плата за гостиницы, занятия спортом, покупка одежды и даже прогулки вдвоем. В каждой из этих сфер вы можете за один раз истратить сумму, представленную не более чем восемью цифрами. Причем компьютер сможет отследить и запомнить до двухсот таких трат.

Первоначальная сумма, которой вы располагаете в начале путешествия, не должна превышать 22-значного числа (в противном случае вам придется приобрести еще несколько таких устройств).

Сразу после внесения очередного бюджетного счета компьютер вычитает из общей суммы произведенные траты и демонстрирует на двустрочном ЖК-дисплее баланс. Всю бухгалтерскую работу устройство может производить с валютой пяти видов.

Кроме того, компьютер двадцать лет может показывать точные год, месяц, дату, день недели, час, минуты, секунды и все 24 часовые зоны мира; помнить паспортные данные владельца, телефонные номера и фамилии и даже выполнять все четыре арифметические операции.

Остается лишь добавить, что «компаньон» весит всего 30 г и питается от собственной литиевой батарейки, что делает его необременительным попутчиком.

Руки вверх, Джексон!

Самая лучшая улика — отпечатки пальцев; ну а если письмо пришло по телефаксу? Вряд ли их удастся обнаружить.

Эра информатики принесла множество сложностей сыщикам. Нетрудно установить, на какой пишущей машинке отпечатан документ, но принтеры имеют значительно меньше индивидуальных особенностей. Текстовый же файл в формате ASCII вообще не имеет особых примет...

Чтобы иметь возможность идентифицировать хотя бы распечатки (пусть только некоторые из них), создана специальная программа. Она предназначена для защиты секретных документов и привносит индивидуальные приметы в распечатку каждого экземпляра документа на лазерном принтере, так что впоследствии можно определить, через чьи руки прошел каждый экземпляр. Приметы (небольшие изменения интервалов между словами и т. п.) сохраняются при ксерокопировании, увеличении, уменьшении, передаче документа по факсу.

Слушай, что я тебе скажу

Устройство распознавания голоса, вмонтированное в автомобильный радиокомплекс Sanyo Electric Tokki, четко распознает 29 приказов, из которых может сформировать для радиоаппаратуры только 20 команд. Но «переучить» прибор реагировать на новые слова можно всего за несколько минут с помощью обычного микрофона.

Может показаться, что такого количества команд для автомобильного радиокомплекса будет многовато. Но это не так, если учесть, что в комплекс входит не только магнитофон, тюнер и усилитель, но и проигрыватель компакт-дисков.

Распознаванием голосовых команд занимается спецпроцессор, который справляется со своей работой уверенно. Даже в условиях движения автомобиля по плохой дороге с большой скоростью электроника «узнает» до 90 процентов предназначенных ей команд, тратя на это всего от 0,2 до 1,5 с.

К сожалению, не сообщается о том, как система относится к салонному разговору, не предназначенному для ее «ушей».

Видеокамера с искусственным интеллектом

Огромное количество технологических новшеств, сделавших видеосъемку простой и доступной практически каждому, до сих пор не сумело вытеснить из почти совершенного устройства вакуумную трубку-видоискатель. Поэтому новый камкордер NV-S2 японской фирмы Matsushita Electric Industrial, оборудованный современным тонкопленочным транзисторным жидкокристаллическим матричным видоискателем (TFT LCD), был с радостью встречен потребительским рынком, несмотря на свою значительную цену 1385 дол.

Цена тем более не покажется высокой, если добавить, что в камере реализован полный комплекс искусственного видеоинтеллекта, реализованного на так называемой фуззи-логике (fuzzy logic). Это и автоматическая наводка на резкость, и стабилизация положения кадра, и автоматическая коррекция контраста изображения, и коррекция цветопередачи при искусственном и дневном освещении, и даже контроль за окружающей температурой и внесение соответствующих поправок.

Характеристики миниатюрного экрана позволяют ему соперничать с лучшими образцами вакуумной техники визуализации: 256 цветов из 16,7 миллиона возможных. А цифровая форма формирования изображения позволяет достаточно просто преобразовать видеосигнал нужным образом и за 80 с получить бумажный оттиск любого кадра с помощью видеопринтера, который, правда, не входит в комплект, но свободно продается рядом.

М. КОВТУН

Программирование в системе Express Pascal

Имена нужно давать не только переменным, но и другим объектам, которые вы описываете в программе (константам, типам, процедурам, функциям и т. п.). Во всех случаях имена строятся по одним и тем же правилам (описанным в предыдущем абзаце). Имя часто называют идентификатором (это слово вы можете встретить в других статьях или книгах, например в документации по системе Express Pascal).

Третье слово — integer — это тип переменной x. Тот факт, что тип переменной x есть integer, означает, что значением переменной может быть целое число в диапазоне от —32768 до +32767.

Двоеточие используется для отделения имени переменной от ее типа; точка с запятой завершает описание переменной.

В других реализациях максимальный размер имени может быть ограничен. Например, компилятор МТ+ требует, чтобы имя было не длиннее 8 литер; Turbo Pascal допускает имена длиной до 63 литер. Возможность использовать длинные имена очень полезна. Пока ваша программа занимает только 6 строк и в ней используется только одна переменная, нетрудно разобраться, для чего используется переменная x. Но когда программа перевалила за 1000 строк, а число переменных — за несколько десятков, им надо давать осмысленные имена. Например, если в переменной нужно хранить счетчик страниц, назовите ее page_counter — и любому (в том числе и вам много месяцев спустя, когда особенности программы будут подзабыты) легко будет понять, что в ней хранится. Конечно, не нужно увлекаться и писать сверхдлинные имена — от этого неоправданно увеличится длина текста, да и вам придется изрядно потрудиться. Например, если в переменной хранится общее число страниц, ее можно назвать number_of_pages, но лучше использовать сокращение, например, назвать ее nr_pages.

Сокращения старайтесь использовать систематически. Например, если вместо слова number где-то использовано nr, делайте так и впредь и никогда не ставьте nr вместо другого слова.

Обратите также внимание на то, как использованы в примерах знаки подчеркивания для разделения длинного имени на отдельные слова. Кроме того, очень удобны для разделения имени прописные буквы. Судите сами: PageCounter, NumberOfPages, NrPages. Но имейте в виду: практически во всех реализациях Паскаля (в частности, в системе Express Pascal) строчные и прописные буквы не различаются (за исключением случаев, когда они входят в текстовые константы), так что имена PageCounter и pagecounter совпадают! Я сам обычно использую прописные буквы в именах процедур и функций, а в именах переменных, констант и типов — знаки подчеркивания.

В первой строке нашей программы есть три слова, выглядящих как имена: var, x и integer. X и integer действительно являются именами: имя x придумали мы, имя integer предопределено в языке для обозначения целочисленного типа. А вот var — это зарезервированное слово. В чем различие между именами и зарезервированными словами? Зарезервированные слова могут употребляться только в одном смысле — в том, который предписывает им синтаксис языка. Слово var обозначает начало описания переменных и не может быть использовано иначе. Имена же можно использовать в любом смысле (там, где требуется имя). Мы назвали переменную x, но могли бы назвать так константу или процедуру. И имя integer мы могли бы использовать для обозначения переменной (обычно этого делать не стоит, чтобы не усложнять работу; есть, однако, случай, когда удобно переопределить предопределенное имя — о нем мы расскажем позже).

В нашей программе использовано три зарезервированных слова: var, begin и end. Всего в Паскале таких слов около

Продолжение. Начало см.: Информатика и образование. 1992. № 1.

50. Мы будем выделять их в тексте **полужирным шрифтом**, чтобы легче было замечать их и привыкать к ним.

Вернемся к тексту программы. (Уже так много сказано, а разобрана только одна строка! Но дальше дело пойдет быстрее.) Он состоит из двух частей: *раздела описаний* и *раздела операторов*. Раздел описаний у нас — одна строка (которую мы разобрали); в ней описывается переменная x.

Раздел операторов всегда начинается зарезервированным словом **begin** (его появление является признаком конца раздела описаний) и заканчивается зарезервированным словом **end**, после которого должна стоять точка. Между **begin** и **end** располагаются *операторы*. Они говорят о том, *что* программа должна делать (а описания — о том, *с чем* программа должна что-то делать). Операторы отделяются друг от друга точкой с запятой.

В нашей программе четыре оператора: первые два вводят число с клавиатуры, следующий вычисляет квадрат числа, последний выводит результат на экран. Займемся сначала оператором, вычисляющим квадрат числа:

```
x := x*x;
```

Его называют *оператором присваивания*. Он состоит из двух частей, левой и правой, соединенных двоеточием и знаком равенства (*знаком присваивания*). В левой части оператора присваивания должна находиться *ссылка на переменную*, в правой — *выражение*.

Ссылка на переменную (в нашем примере это просто имя переменной; о других видах ссылок на переменную мы поговорим позднее) указывает на место в памяти («ящичек»), в которое можно что-то записать.

Выражение строится из констант, ссылок на переменные и *вызовов функций* с помощью знаков операций и круглых скобок по обычным математическим правилам. В арифметических выражениях для целых чисел можно использовать знаки

```
+       сложение
-       вычитание
*       умножение
div     деление нацело
mod     взятие остатка
```

(внимание! Некоторые знаки операций являются зарезервированными словами: здесь это **div** и **mod**). Порядок выполнения операций — обычный математический: сначала умножение и деление, потом сложение и вычитание. Если порядок нужно изменить, употребляются скобки.

Константы в целочисленных арифметических выражениях записываются как обычные десятичные числа: 12, 1, 5421 и т. д. Пробелы внутри констант недопустимы.

Примеры выражений:

```
a + b
23 + x*y
-(nr_lines-1) div 60 - (x1+y1) mod z22
```

Выполняется оператор присваивания так: вычисляется выражение, стоящее в правой части, и полученное значение присваивается переменной, указанной в левой части (т. е. кладется в ящичек, на который указывает ссылка на переменную в левой части). Использование ссылок на переменные в левой и правой частях существенно различается. У переменной, находящейся в правой части, нас интересует только значение — содержимое ящичка; нам совершенно неважно, где этот ящичек находится. У переменной из левой части нас, наоборот, интересует только ее местонахождение — в этот ящичек должно быть помещено вычисленное значение, а исходное содержимое неважно — оно просто теряется в результате выполнения оператора присваивания.

В нашей программе при присваивании берется значение переменной x (а туда оператор Readln(x) поместил то, что вы ввели с клавиатуры) и умножается опять же на значение переменной x. После того как значение вычислено, оно помещается в переменную x. Запомните: сначала вычисляется выражение: если в левой и правой частях оператора присваивания указана одна и та же переменная, то при вычислении выражения будет использоваться исходное значение переменной и только потом ей будет присвоено новое значение.

Немного подробнее об операциях **div** и **mod**. Для положительных чисел их применение дает очевидный результат:

25 **div** 6=4		25 **mod** 6=1	
17 **div** 3=5		17 **mod** 3=2	
15 **div** 5=3		15 **mod** 5=0	

Если же один или оба операнда отрицательны, то результат операции i **div** j есть математическое частное от деления i на j, округленное до ближайшего целого значения по направлению к 0, а результат операции **mod** определяется по формуле i **mod** j=i—(i **div** j)j. Таким образом, знак результата операции **mod** всегда совпадает со знаком левого операнда. Примеры:

25 **div** —6=—4		25 **mod** —6=1	
—25 **div** 6=—4		—25 **mod** 6=—1	
—25 **div** —6=4		—25 **mod** —6=—1	
17 **div** —3=—5		17 **mod** —3=2	
—15 **div** 5=—3		—15 **mod** 5=0	

Теперь займемся операторами, расположенными в первой и последней строках нашей программы. Они выполняют вывод информации на экран и ввод информации с клавиатуры. По своей форме (синтаксически) эти операторы являются *вызовами процедур*. Оператор вызова процедуры состоит из имени процедуры и списка параметров, заключенного в скобки.

Процедура — это набор действий, выполнять которые нужно достаточно часто; чтобы в программе не появлялись постоянно одни и те же куски, эти действия описываются один раз и оформляются в виде процедуры. Теперь для их выполнения достаточно вызвать процедуру с помощью соответствующего оператора вызова процедуры.

Процедуры для своей программы обычно пишет сам программист. Есть, однако, наборы действий, встречающиеся почти во всех программах — например, ввод и вывод данных. Для выполнения таких общих действий предназначены *стандартные процедуры*. Они создаются разработчиками компиляторов Паскаля, и их не нужно описывать — вы можете просто пользоваться ими. В языке Express Pascal около 90 стандартных процедур.

Рассмотрим четыре стандартные процедуры: Read, Readln, Write, Writeln. Они выполняют ввод с клавиатуры и вывод на экран. Эти процедуры имеют переменное число параметров (может быть, и ни одного).

Параметрами Write являются выражения. Процедура преобразует их в текстовый вид и выводит на экран. В первом вызове процедуры Write нашего примера один параметр — строка «Введите число: ». Строковый параметр выводится на экран без всякого преобразования, и вы получаете в результате на экране строку «Введите число: ». Вывод на экран начинается с текущей позиции (при старте программы она находится в левом верхнем углу экрана; в текущей позиции находится курсор); после окончания вывода текущей становится та позиция, где закончился вывод текста. Если в процессе вывода достигается конец строки, то вывод продолжается с начала следующей (например, если конец строки достигнут в процессе вывода числа, то оно окажется разорванным на две части).

В последней строке нашей программы вызывается процедура Writeln с двумя параметрами. Первый из них — строка, второй — переменная x. Переменная x имеет тип integer, поэтому ее содержимое будет выводиться в формате целого числа. Вывод на экран текстового представления числа, записанного в x, начнется сразу же после окончания предыдущего текста, никаких пробелов вставлено не будет. Поэтому мы и поставили пробел в конце строковой константы «Квадрат введенного числа= ».

Writeln отличается от Write только тем, что она после завершения вывода переводит текущую позицию в начало следующей строки.

Чтобы лучше понять, как работают процедуры Write и Writeln, попробуйте выводить с их помощью различный текст на экран и посмотрите, что у вас будет получаться.

Процедура Read выполняет ввод с клавиатуры текстов и чисел. Ее параметры — ссылки на переменные. Входная информация должна быть оформлена по определенным правилам в соответствии с типом переменной. В нашем примере параметр — переменная x типа integer, поэтому должна быть введена последовательность цифр, представляющая целое число, возможно, начинающаяся со знака «+» или «—». Если же

будет введено что-либо другое, возникнет ошибка.

Когда вы, работая с программой, выполняете ввод, инициированный процедурой Read, все вводимые символы немедленно отображаются на экране. Но в программу эта информация попадет только после того, как вы нажмете клавишу «возврат каретки». Поэтому можно исправлять допущенные при вводе ошибки — нажатие клавиши back space (она называется еще «возврат на шаг», находится в правом верхнем углу клавиатуры и обозначена двойной стрелкой влево) приводит к стиранию последнего введенного символа.

Числа вводятся по тем же правилам, по которым они записываются в программе. Целые числа записываются последовательностью десятичных цифр, возможно, со знаком «+» или «—» в начале.

С помощью одного вызова процедуры Read можно ввести несколько чисел. При вводе они могут разделяться пробелами или располагаться на разных строках. Если в строке вы ввели больше чисел, чем задано параметров, то оставшуюся часть строки программа запомнит и использует при следующем вызове процедуры Read. Например, если в программе написано

```
Read(x); Read(y); Read(z);
```

(переменные x, y, z типа integer), а вы ввели с клавиатуры

```
11  222    3333
```

то при выполнении первого вызова процедуры Read будет прочитано первое число (11) и занесено в переменную x; остаток будет использован при выполнении следующего вызова процедуры Read, и в переменную y попадет значение 222; в переменную z попадет значение 3333.

Readln отличается от Read тем, что после завершения ввода объектов, заданных ее параметрами, она пропускает все оставшиеся в строке символы до конца строки. Поэтому, если в программе написано

```
Readln(x); Readln(y); Readln(z);
```

а вы ввели с клавиатуры

```
11  222    3333
```

то при выполнении первого вызова процедуры Readln будет прочитано первое число (11) и занесено в переменную x, остаток же будет отброшен. При выполнении следующего вызова процедуры Readln программа снова будет ожидать ввода с клавиатуры. Таким образом, вам придется ввести три строки, и из каждой будет использовано только первое число.

В заключение нашего простого примера рассмотрим другие целочисленные типы. Переменная x имеет у нас тип integer, это значит, что значениями ее могут быть целые числа в диапазоне от —32768 до +32767. Имеется еще четыре целочисленных типа:

shortint — его значениями могут быть целые числа в диапазоне от —128 до +127;

longint — его значениями могут быть целые числа в диапазоне от —2147483648 до +2147483647;

byte — его значениями могут быть целые числа в диапазоне от 0 до 255;

word — его значениями могут быть целые числа в диапазоне от 0 до 65535 (те, кто знаком с двоичной системой счисления и с байтами, сразу увидят, что это значения, которые можно разместить в одном, двух или четырех байтах). Зачем же столько целочисленных типов? Почему бы не пользоваться только типом longint? Из соображений экономии. Чем короче значение, тем меньше места будет занимать скомпилированная программа и тем быстрее будет она работать. Если вас не волнуют размер программы и ее быстродействие, можете спокойно забыть про все целочисленные типы, кроме longint; но если вам важна эффективность программы, выбирайте тип покороче.

Лучший способ научиться программировать — писать программы. Попробуйте, как сумеете, изменить наш простой пример: пусть вычисляется не квадрат введенного числа, а какое-нибудь другое значение; пусть вводится не одно, а два числа и вычисляются их сумма и произведение; пусть выводятся на экран разные строки и числа. Пробуйте!

Продолжение следует.

Е. БАКМАН, И. ДРЕМОВА
Черниговский педагогический институт

Введение в микро-Пролог

Логические операции

Вся математическая логика построена на логических операциях конъюнкции, дизъюнкции, отрицания, импликации и др., которые могут быть выражены через основные. Так как Пролог основан на логике предикатов, он также не может обойтись без основных операций.

С конъюнкцией мы уже встречались. Условия в правилах являются конъюнкциями подцелей. Конъюнкция не имеет специального обозначения, она подразумевается синтаксическими правилами.

Выясним, что представляет собой дизъюнкция. Для примера определим отношение «быть родителем» через отношения «быть матерью» и «быть отцом». А именно: X является родителем Y, если X является отцом Y или X является матерью Y.

В этом предложении «X является родителем Y» — заключение, а условие состоит из двух предикатов: «X является отцом Y», «X является матерью Y», соединенных союзом «или».

Очевидно, заключение будет верным, если будет верным хотя бы один из предикатов условия.

Таким образом, предложение **A или B** (где A, B — предикаты, а могут быть и предложения) считают истинным во всех случаях, когда хотя бы один (т. е. один или оба) из предикатов A или B являются истинными. Такое предложение, составленное из двух и более предикатов или предложений с помощью союза «или», называется дизъюнкцией.

В Прологе дизъюнкция обозначается предикатным символом OR, за которым следует любое число аргументов — предложений, являющихся конъюнкциями предикатов.

Рассмотренное выше отношение «быть родителем» в Прологе запишется так:

```
_.ADDCL (( РОДИТЕЛЬ X Y )
   ( OR (( ОТЕЦ X Y )) (( МАТЬ X Y ))))
```

Есть в Прологе и отрицание NOT. Для отношений над числами оно совпадает с привычным для нас отрицанием не. Аргумент у NOT один: либо предложение, являющееся конъюнкцией входящих в него предикатов, либо просто один предикат. Например, NOT EQ X 2 означает отношение X<>2, а NOT SUM 5 X Y X+5<>Y, если X и Y уже присвоены какие-то значения. Точнее, они обязательно должны быть присвоены, иначе может получиться перебор слишком большого числа различных значений. Ведь существует бесконечно много X, удовлетворяющих неравенству X<>2.

По этой причине перед негативным условием вида (NOT предикат) должно идти либо позитивное условие, либо значения переменных должны фиксироваться в заключении. Например, ((БЕЗДЕТЕН X) (NOT РОДИТЕЛЬ X Y)) означает, что X бездетен, если не существует в базе знаний такого Y, что X родитель Y. Здесь имеется в виду, что в запросе будет указано конкретное значение X.

Употребление NOT с числовыми отношениями для нас привычно, так как школьное математическое образование привило нам двузначную логику: относительно любого утверждения можно сказать, что оно либо истинно, либо ложно; третьего не дано.

Пролог наглядно демонстрирует ограниченность такого подхода. Если сделать запрос (ПОСАДИЛ ДЕД ЛУК), то положительный ответ на него не будет найден. Означает ли это, что дед не посадил лук? Конечно, нет. Просто в базе знаний нет сведений об этом. Следовательно, ответом будет неопределенность.

Это говорит об относительности знаний. В любой момент могут поступить новые сведения и неопределенность сменится определенностью. Но никогда знания не будут полными. Всегда найдутся такие запросы, ответ на которые будет неопределенным.

Окончание. Начало см.: Информатика и образование. 1991. № 6; 1992. № 1.

В Прологе для нечисловых отношений принято: истинность выражения (NOT условие) означает, что в базе знаний нет сведений, подтверждающих истинность указанного условия. Если же такие сведения есть, то негативное условие принимает значение «ложно». В результате получается: NOT верно=неверно, NOT нет сведений =верно.

А как же насчет «ложно»? Такого понятия в Прологе нет. Если есть информация отрицательного характера, то ее нужно вводить в базу знаний соответствующим негативным отношением, например НЕ—ЯВЛЯЕТСЯ—ЕЖОМ ЧЕРЕПАХА.

Рассмотрим следующий пример: характерной особенностью птиц является умение летать. В то же время некоторые виды птиц не летают, в частности не летают пингвины и страусы.

Объясним все это Прологу с помощью отношений «быть птицей» (ПТИЦА X) и «не уметь летать» (НЕ—ЛЕТАЕТ X). Введем факты о различных птицах:

```
_.ADDCL ((ПТИЦА ГОЛУБЬ))
_.ADDCL ((ПТИЦА ПЕЛИКАН))
_.ADDCL ((ПТИЦА ЧАЙКА))
_.ADDCL ((ПТИЦА СТРАУС))
_.ADDCL ((ПТИЦА ПИНГВИН))
```

Теперь сообщим о нелетающих птицах:

```
_.ADDCL ((НЕ-ЛЕТАЕТ СТРАУС))
_.ADDCL ((НЕ-ЛЕТАЕТ ПИНГВИН))
```

Используя отношения ПТИЦА и НЕ—ЛЕТАЕТ, мы можем выяснить, летает ли голубь. Правило такое: если это птица и нет сведений, что она не летает, то она летает (наследование свойства при умолчании).

```
_.ADDCL ((ЛЕТАЕТ X) (ПТИЦА X)
                    (NOT НЕ-ЛЕТАЕТ X))
```

Зададим вопрос:

```
_.КАКИЕ (X (ЛЕТАЕТ X))
ГОЛУБЬ
ПЕЛИКАН
ЧАЙКА
_.
```

Здесь NOT использовано в том самом непривычном смысле, о котором шла речь выше. NOT НЕ—ЛЕТАЕТ X верно, если нет сведений, что X не летает, т. е. NOT неопределенность=верно.

Чувствуете, насколько неполно наше описание действительности? Например, с пеликаном. Он попал в список летающих потому, что нет сведений о том, что он не летает. А вдруг он не летает, а мы не сообщили об этом Прологу?

Если сделать запрос «верно ли, что аист летает?» (как это сделать, описано ниже), Пролог ответит «не знаю», и это будет прекрасный ответ: ведь мы действительно не сообщили, является ли аист птицей.

А как быть с бабочками? Они хоть и не птицы, но тоже летают. Чем больше мы сообщаем сведений Прологу, тем больше появляется сопутствующих вопросов, на которые будет ответ «не знаю». Это похоже на парадокс: пополняя базу знаний, мы усугубляем ее неполноту.

Подобное явление обнаружил Сократ более двух тысяч лет назад. Он сказал своему рабу примерно следующее: «Мое незнание больше твоего, так как я знаю больше».

Подведем итог сказанному выше.

Отрицание NOT в Прологе означает отсутствие сведений. Введение негативного знания позволяет сделать более точный вывод, но полностью не снимает неопределенность.

Появляются три возможности:

утверждение верно;

негативное утверждение верно;

пункты 1 и 2 не выполняются (неопределенность).

NOT <утверждение> верно, если встретился второй или третий случай.

Управление процессом вывода

Алгоритм вывода (проверки истинности предиката) очень похож на команду выбора:

```
ВЫБОР
    ПРИ  P11 И P12 И ... И P1n1 :
    ПРИ  P21 И P22 И ... И P2n2 :
    . . . . . . . . . .
    ПРИ  Pm1 И Pm2 И ... И PmPm :
ВСЕ
```

Здесь через P_{ij} обозначена j-я подцель i-го правила, содержащего требуемое отношение. Как видно из этой за-

писи, область серий в команде выбора не заполнена.

Причина в том, что в Прологе нет команд в явном виде. Конечно, без процедурных элементов обойтись нельзя, просто они для единообразия имеют вид предикатов и расположены в области условий (подцелей).

Алгоритм работы системы Пролога следующий: выбирается очередное правило, и последовательно слева направо проверяются его подцели. Если для проверки подцели нужна проверка правил с новым отношением, то это рассматривается как переход к вспомогательному алгоритму. Предположим, что какая-то подцель приняла значение «неверно». Тогда пробуются другие возможные значения переменных. И только когда все возможности применения данного правила исчерпаны, осуществляется переход к следующему правилу.

В целом результат запроса будет «неверно», если не удалось применить ни одного правила.

Как видим, процесс проверки очень трудоемкий и занимает много времени. С целью экономии машинного времени, а также для управления процессом проверки истинности формул используется знак /, который рассматривается как всегда истинный предикат, но он воздействует на работу исполнительной системы Пролога.

После его обработки запрещается переход к другим правилам данного отношения.

Рассмотрим работу предиката / на примере.

```
_.ADDCL ((НЕВЕРНО X) X / FAIL)
_.ADDCL ((НЕВЕРНО X))
```

Аналогично определено встроенное отношение NOT.

Для проверки истинности НЕВЕРНО X, где X — предикат, сначала применяется первое правило, в котором в качестве первой подцели стоит сам предикат X. Если он дает значение «неверно», то происходит переход ко второму правилу, которое всегда выполняется успешно. Например,

```
_. НЕВЕРНО (VAR 5)
_.
```

Результат запроса можно понять так: «Да, неверно, что 5 — переменная».

Теперь предположим, что значение X «верно». Следом идут косая черта, которая запрещает переход ко второму правилу, и предикат FAIL, который никогда не удовлетворяется. Поэтому результатом будет «неверно» и должен печататься вопросительный знак.

Убедимся в сказанном на примере.

```
_.ADDCL ((ОТЕЦ ПАПА-КАРЛО БУРАТИНО))
_.НЕВЕРНО (ОТЕЦ ПАПА-КАРЛО БУРАТИНО)
?
```

Попробуем обойтись без / в определении отношения НЕВЕРНО.

```
_.KILL НЕВЕРНО
_.ADDCL ((НЕВЕРНО X) X FAIL)
_.ADDCL ((НЕВЕРНО X))
_.НЕВЕРНО (ОТЕЦ ПАПА-КАРЛО БУРАТИНО)
_.
```

Теперь знак вопроса не печатается, так как выполняется второе правило. Но так не должно быть. Если же второе правило убрать, то неправильно будут выполняться запросы, в которых X неверно.

Таким образом, мы убедились в полезности предиката /.

Унификация

При попытке применить некоторое правило для проверки истинности предиката требуется сопоставить списки их аргументов. Если сопоставление проходит успешно и переменным присваиваются соответствующие им значения, то правило применяется, иначе резуль-

татом является неудача. Процесс сопоставления называется унификацией.

Рассмотрим пример. Пусть в качестве аргумента в правиле стоит (X |), а в предикате — A B C). В этом случае сопоставление проходит успешно и в результате получается X=A, Y= =(B C).

Теперь пусть в правиле аргумент равен (X X C), а в предикате (A Y Y). Унификация приводит к неудаче — не удается подобрать значения переменных X и Y так, чтобы списки совпадали.

Единственной задачей предиката EQ X Y является попытка сопоставить первый и второй аргументы. Так, EQ X 2 приводит к удаче, если положить X=2, а EQ A B — к неудаче. С помощью EQ можно проверить и предыдущие примеры:


```
_.?(EQ (X!Y) (A B C))
_*
```

Сопоставление прошло успешно.

```
_.?(EQ (X X C) (A Y Y))
?
_*
```

Напечатан вопросительный знак. Значит, сопоставление не удалось.

Если сопоставление удачно, можно напечатать его результат.

```
_.?((EQ (X!Y) (A B C)) (PP Y))
  (B C)
_*
```

Как видим, Y — это список, состоящий из всех элементов, остающихся после исключения первого элемента A.

Форма будущего запроса должна учитываться при записи правил и фактов, так как невозможность сопоставления аргументов отношения приводит к неудаче. Например, при запросе

```
_.КАКИЕ (X ( ПТИЦА (X)))
```

не будет получено ни одного имени птицы — сопоставление (X) с ГОЛУБЬ приводит к неудаче, так как ГОЛУБЬ не является списком.

Встроенное отношение NOT определено с помощью предложений

```
((NOT!X) X / FAIL)
((NOT!X))
```

Попытаемся по этой записи определить допустимые синтаксические конструкции с данным отношением.

Так как после NOT стоит |X, то это любая последовательность термов, т. е. отношение NOT многоаргументное. Далее в качестве подцели стоит X, что означает: проверить X на истинность. Теперь X — это список (так как нет вертикальной черты). Такой запрос возможен, если X — это один предикат в скобках, |X — то же, но без скобок. Следовательно, унификация будет удачной, например, для (NOT БРАТ Y КОЛЯ). Тогда X=(БРАТ Y КОЛЯ), что является допустимым запросом.

Запрос

```
(NOT (LESS 3 2))
```

приводит к сообщению об ошибке

```
error: 5   goal: ((LESS 3 2))
```

которое означает: «Неверная подцель ((LESS 3 2))».

В этом случае X=((LESS 3 2)), что является недопустимой подцелью. В то же время (NOT LESS 3 2) дает «верно». Здесь |X=LESS 3 2, X=(LESS 3 2), а это правильно записанная подцель. Следовательно, после NOT должен идти предикат без скобок.

В естественном языке тоже используется сопоставление. В отличие от Пролога совпадающими по смыслу являются различные словоформы одного слова, например семья и семью. Иногда это может привести к ошибочным сопоставлениям, например слово «семью» может быть сопоставлено и со словом «семь» (сравните также «суда» от «суд» и от «судно»). Кроме того, в русском языке произволен и порядок слов в предложении, что оставляет еще меньше ограничений на возможность сопоставления.

Пример простейшей экспертной системы

Машинная программа, которая владеет набором факторов и правил из специальной области знаний и которая в состоянии манипулировать этой информацией, называется экспертной системой. Пролог предоставляет прекрасные воз-

можности для построения экспертных систем.

Для построения экспертной системы требуется получить знания у эксперта, что является самым трудным этапом разработки. Дело в том, что в большинстве предметных областей знаний не формализованы, а их использование происходит на интуитивном уровне.

Разработка экспертной системы приносит двойную пользу, так как при этом знания четко систематизируются, расплывчатые понятия принимают строгие очертания. Это позволяет с большей эффективностью обучать не только ЭВМ, но и людей.

В качестве примера мы рассмотрим взаимосвязи таких понятий, как *знания, сведения, грамотность, невежественный, дилетант* и др. В качестве отношений выберем: синоним, антоним, имеет, разновидность.

Чтобы экспертная система не была слишком сложной, будем делать некоторые упрощения. Так, например, понятия *знания, сведения, информация* будем считать синонимами, хотя между ними есть и оттенки различий. Пусть *знания* будет главным синонимом. Зафиксируем это с помощью отношения ГЛАВСИН:

```
((ГЛАВСИН ЗНАНИЯ СВЕДЕНИЯ))
((ГЛАВСИН ЗНАНИЯ ИНФОРМАЦИЯ))
((ГЛАВСИН ЗНАНИЯ ЗНАНИЯ)).
```

Последнее предложение означает, что ЗНАНИЯ будет главным синонимом и для самого слова *знания*.

Два слова X и Y являются синонимами, если для них существует хоть один общий главный синоним Z:

```
((СИНОНИМ X Y)(ГЛАВСИН Z X)
                    (ГЛАВСИН Z Y))
```

Теперь мы уже сможем находить все синонимы слова с помощью КАКИЕ. Например:

```
_. КАКИЕ (X (СИНОНИМ ЗНАНИЯ X))
СВЕДЕНИЯ
ИНФОРМАЦИЯ
ЗНАНИЯ
_.
```

Следующая группа понятий связана с отношением ИМЕЕТ (имеется в виду обладание знаниями).

```
      ((ИМЕЕТ КОМПЕТЕНТНЫЙ ЗНАНИЯ))
*     ((ИМЕЕТ ГРАМОТНЫЙ ЗНАНИЯ))
```

Как видим, *компетентный* и *грамотный* — синонимы, поэтому определим

```
((ГЛАВСИН КОМПЕТЕНТНЫЙ ГРАМОТНЫЙ))
((ГЛАВСИН КОМПЕТЕНТНЫЙ КОМПЕТЕНТНЫЙ))
```

Теперь определим отношение НЕ—ИМЕЕТ:

```
      ((НЕ-ИМЕЕТ НЕКОМПЕТЕНТНЫЙ ЗНАНИЯ))
*     ((НЕ-ИМЕЕТ НЕВЕЖЕСТВЕННЫЙ ЗНАНИЯ))
*     ((НЕ-ИМЕЕТ БЕЗГРАМОТНЫЙ ЗНАНИЯ))
```

Снова имеем группу синонимов:

```
((ГЛАВСИН НЕКОМПЕТЕНТНЫЙ
                    НЕВЕЖЕСТВЕННЫЙ))
((ГЛАВСИН НЕКОМПЕТЕНТНЫЙ
                    БЕЗГРАМОТНЫЙ))
((ГЛАВСИН НЕКОМПЕТЕНТНЫЙ
                    НЕКОМПЕТЕНТНЫЙ))
```

Так как предполагается, что синонимы тождественны, то, чтобы не повторять с ними одни и те же отношения, введем новый запрос.

```
_.ADDCL ((КАКИЕСИН (X!Y))
          (КАКИЕ (Z (СИНОНИМ Z X) !Y)))
```

Тогда предложения, помеченные слева звездочкой, можно не вводить в базу знаний.

```
_.КАКИЕСИН (X (ИМЕЕТ X ЗНАНИЯ))
  КОМПЕТЕНТНЫЙ
  ГРАМОТНЫЙ
```

Рассмотрим теперь понятия *догма, предрассудок, вера*. Все это разновидности знаний, включая науку. Зафиксируем это:

```
((РАЗНОВИДНОСТЬ ДОГМА ЗНАНИЯ))
((РАЗНОВИДНОСТЬ ПРЕДРАССУДОК ЗНАНИЯ))
((РАЗНОВИДНОСТЬ ВЕРА ЗНАНИЯ))
((РАЗНОВИДНОСТЬ НАУКА ЗНАНИЯ))
```

Науку от любых сведений или знаний отличает наличие целей — предсказание и объяснение. Итак,

```
((ЦЕЛЬ ПРЕДСКАЗАНИЕ НАУКА))
((ЦЕЛЬ ОБЪЯСНЕНИЕ НАУКА))
```

Чтобы знать, чем различаются между собой разновидности знаний, дадим им определения:

```
((ОПРЕДЕЛЕНИЕ ДОГМА) (РР ДОГМА – ЭТО
                    ЗАСТЫВШЕЕ ЗНАНИЕ))
((ОПРЕДЕЛЕНИЕ ПРЕДРАССУДОК) (РР
    ПРЕДРАССУДОК – ЭТО ЛОЖНОЕ ЗНАНИЕ))
((ОПРЕДЕЛЕНИЕ ВЕРА) (РР ВЕРА – ЭТО
```

```
                ЗНАНИЕ, ПРИНИМАЕМОЕ БЕЗ
                        ДОКАЗАТЕЛЬСТВА))
((ОПРЕДЕЛЕНИЕ НАУКА) (РР НАУКА - ЭТО
      ЗНАНИЯ, ИМЕЮЩИЕ ЦЕЛЬЮ ОБЪЯСНЕНИЕ
                        И ПРЕДСКАЗАНИЕ))
```

Последние четыре предложения по сути являются просто командами печати определений. Например:

```
_.ОПРЕДЕЛЕНИЕ ДОГМА
 ДОГМА - ЭТО ЗАСТЫВШЕЕ ЗНАНИЕ
_.
```

С рассматриваемой нами предметной областью связано и слово *дилетант*. Дилетант имеет поверхностные знания. Обратите внимание на то, что некоторые разновидности знаний имеют свой термин для обозначения (например, застывшее знание — догма), а некоторые нет (например, поверхностные знания). В таких случаях к основному термину добавляется прилагательное или существительное в родительном падеже (например, дверная ручка и ручка двери).

Мы не будем вводить новый термин для поверхностных знаний:

```
((РАЗНОВИДНОСТЬ ПОВЕРХНОСТНЫЕ-ЗНАНИЯ
                        ЗНАНИЯ))
((ИМЕЕТ ДИЛЕТАНТ ПОВЕРХНОСТНЫЕ-ЗНАНИЯ))
```

Точно так же мы можем определить, что верующий — тот, кто имеет веру:

```
((ИМЕЕТ ВЕРУЮЩИЙ ВЕРА))
```

Определим еще отношение «антоним». Для этого определим сначала ГЛАВ—АНТ — главный антоним — отношение, определяющее антонимичность главных синонимов.

```
((ГЛАВ-АНТ КОМПЕТЕНТНЫЙ
                        НЕКОМПЕТЕНТНЫЙ))
```

Тогда антоним:

```
((АНТОНИМ X Y) (ГЛАВСИН X1 X)
  (ГЛАВСИН Y1 Y)(OR ((ГЛАВ-АНТ X1 Y1))
((ГЛАВ-АНТ Y1 X1))))
```

То есть два термина являются антонимами, если их главные синонимы находятся в отношении ГЛАВ—АНТ.

Теперь можно легко находить слова, состоящие в заданном отношении друг с другом.

Предположим, нам нужны все разновидности зданий.

```
_.КАКИЕ (X (РАЗНОВИДНОСТЬ X ЗНАНИЯ)
ДОГМА
ПРЕДРАССУДОК.ВЕРА
ПОВЕРХНОСТНЫЕ-ЗНАНИЯ
НАУКА
_.
```

Теперь уточним значение слова *предрассудок*.

```
_.ОПРЕДЕЛЕНИЕ ПРЕДРАССУДОК
 ПРЕДРАССУДОК - ЭТО ЛОЖНОЕ ЗНАНИЕ
_.
```

Найдем антонимы слова *компетентный*.

```
_.КАКИЕ (X (АНТОНИМ X КОМПЕТЕНТНЫЙ))
 НЕВЕЖЕСТВЕННЫЙ
 БЕЗГРАМОТНЫЙ
 НЕКОМПЕТЕНТНЫЙ
_.
```

Какова цель науки?

```
_.КАКИЕ (X (ЦЕЛЬ X НАУКА))
 ПРЕДСКАЗАНИЕ
 ОБЪЯСНЕНИЕ
_.
```

Что такое наука?

```
_.ОПРЕДЕЛЕНИЕ НАУКА
 НАУКА - ЭТО ЗНАНИЯ, ИМЕЮЩИЕ ЦЕЛЬЮ
                ОБЪЯСНЕНИЕ И ПРЕДСКАЗАНИЕ
_.
```

Оказывается, значение слова *наука* полностью задано отношениями, в которых оно состоит с другими словами: наука — это разновидность знания; наука имеет две цели — объяснение и предсказание. Но это то же самое, что сказано в определении.

Итак, значение термина не более чем перечень всех отношений, в которых данный термин находится с другими понятиями (нечто напоминающее круговую поруку).

Полученный словарь представляет собой тезаурус (так называют словари, содержащие списки используемых терминов и взаимные связи между ними для какой-либо конкретной предметной области, т. е. именно то, что мы ввели в базу знаний).

Тезаурус определяет систему понятий настолько строго, насколько это вообще возможно. Наиболее важным практическим применением тезауруса является поиск ассоциаций — всех слов,

находящихся в непосредственной связи с данным словом. Но Пролог ведет поиск только по заданным отношениям. А как быть, если отношения заранее неизвестны? Есть выход и в таком случае.

Все введенные нами имена Пролог заносит в отношение DICT (от dictionary — словарь). Все эти имена располагаются после третьего аргумента. Если записать

```
(DICT * X1 Y !Z)
```

то Z как раз и будет представлять собой список всех имен. Правда, среди них будут не только имена отношений.

Однако нам потребуются имена отношений по одному, а не списком. Для этого с помощью двух правил рекурсивно определим отношение В—СПИСКЕ.

```
((В-СПИСКЕ X (X!Y)))
```

— X является элементом списка, если X — первый элемент этого списка:

```
((В-СПИСКЕ X (Y!Z)) (В-СПИСКЕ X Z))
```

— X является элементом списка, если X является элементом оставшейся части списка.

Теперь можно определить отношение СЛОВАРЬ с помощью следующего правила:

```
((СЛОВАРЬ X) (DICT * X1 Y !Z)
                    (В-СПИСКЕ X Z))
```

Оно означает: СЛОВАРЬ X верно, если X находится в списке имен Z. Так, например, запрос КАКИЕ (X (СЛОВАРЬ X)) напечатает все введенные нами имена.

Определим отношение «ассоциация»:

```
((АССОЦИАЦИЯ X Y Z )(СЛОВАРЬ Z)
          (OR ((Z X Y)) ((Z Y X ))) )
```

Оно означает, что X является ассоциацией Y, если найдется в словаре такое отношение Z, для которого X и Y являются аргументами. Например,

```
_.КАКИЕ (X (ДОГМА X ЗНАНИЯ))
 РАЗНОВИДНОСТЬ
_.
```

Мы спросили: «Какая связь между догмой и знаниями?» Ответ: «Догма — разновидность знаний».

Итак, поставленная цель поиска ассоциаций достигнута. Однако такого рода запросы выполняются долго, так как в качестве отношения Z пробуются все имена базы знаний, причем среди них не все являются отношениями. Человек при поиске ассоциаций не перебирает все имена, поэтому он находит нужное слово быстрее, хотя и у людей иногда возникают трудности при вспоминании.

Ускорить поиск ассоциаций можно путем введения тематических словарей, например словаря отношений, связанных со знаниями, а также словаря универсальных отношений, общих для всех тематических словарей. Примеры универсальных отношений:

```
((УНИВ-ОТН СИНОНИМ))
((УНИВ-ОТН РАЗНОВИДНОСТЬ))
((УНИВ-ОТН АНТОНИМ))
((УНИВ-ОТН ИМЕЕТ))
```


В нашем тезаурусе все отношения универсальны. И это неудивительно, так как область знаний — одна из самых общих областей. Примеры тематических (родственных) отношений:

```
((РОДСТВ-ОТН РОДИТЕЛЬ))
((РОДСТВ-ОТН МАТЬ))
((РОДСТВ-ОТН ДОЧЬ))
```

Перечислить нужно все родственные отношения, которые вы захотите проверить. Тогда ускоренно работающее отношение «родственная ассоциация» определится так:

```
((РОДСТВ-АССОЦ X Y Z) (OR
      ((РОДСТВ-ОТН Z)) ((УНИВ-ОТН Z)))
                          (Z X Y ))
```

Предположим, что нам требуется найти всех родственников Ольги:

```
_.КАКИЕ ((ОЛЬГА Z Y)
            (РОДСТВ-АССОЦ ОЛЬГА Y Z))
  (ОЛЬГА РОДИТЕЛЬ ВЕРА)
  (ОЛЬГА МАТЬ ВЕРА)
  (ОЛЬГА ДОЧЬ ПЕТР)
_.
```

А теперь то же для Веры:

```
_.КАКИЕ ((ВЕРА Z Y)
            (РОДСТВ-АССОЦ ВЕРА Y Z))
  (ВЕРА ДОЧЬ ОЛЬГА)
  (ВЕРА РАЗНОВИДНОСТЬ ЗНАНИЯ)
```

Неожиданно мы столкнулись с явлением омонимии, при котором одно и то же слово имеет несколько значений. В

данном случае Вера — это женское имя и одновременно одна из разновидностей знаний.

Заключение

Существует много версий Пролога. Так, для IBM-совместимых машин есть Турбо-Пролог, для «Ямахи» кроме рассмотренного микро-Пролога существует Пролог-Д. Различия между языками зачастую чисто внешние, касаются лишь синтаксиса. Например, вместо (ЖЕНА АННА НИКОЛАЙ) может быть запись ЖЕНА (АННА НИКОЛАЙ). Между подцелями могут стоять запятые, заключение от подцелей отделяется знаками: — или < —. Есть отличия и в терминологии. В любом случае тот, кто освоил одну из версий Пролога, легко освоит и остальные.

Литература

1. *Кларк К., Маккейб Ф.* Введение в логическое программирование на микро-Прологе. М.: Радио и связь, 1987.
2. *Клоксин У., Меллиш К.* Программирование на языке Пролог. М.: Мир, 1987.
3. *Хоггер К.* Введение в логическое программирование. М.: Мир, 1988.
4. Логическое программирование / Ред. В. Н. Агафонова. М.: Мир, 1988.
5. Язык Пролог в пятом поколении ЭВМ: Сб. статей / Сост. Н. И. Ильинский. М.: Мир, 1988.
6. *Григорьев С. Г., Морозов М.* Давайте попробуем Пролог // Информатика и образование. 1987. № 4.
7. *Каймин В. А., Григорьев С., Угринович Н., Щеголев А.* Элементы логики и начала языка Пролог // Информатика и образование. 1989. № 4.
8. *Каймин В. А., Григорьев С. Г.* Пролог в школьной информатике // Информатика и образование. 1990. № 2.
9. Логическое программирование. М.: Знание, 1988. (Сер. «Выч. техн., ее применение», № 9).
10. *Доорс Дж., Рейблейн А. Р., Вадера С.* Пролог — язык программирования будущего. М.: Финансы и статистика, 1990.
11. *Стерлинг Л., Шапиро Э.* Искусство программирования на языке Пролог. М.: Мир, 1990.
12. *Иванова Г С., Тихонов Ю. В.* Введение в язык МПролог. М.: Изд-во МГТУ, 1990.

Научно-методический центр «Агат»

Шесть лет назад началось массовое внедрение компьютеров в школу. Кому-то повезло больше, и школы получили импортные ЭВМ, кому-то меньше — у них ЭВМ отечественные. Сейчас мы наблюдаем два процесса: «старение» школьных компьютеров и попытки ввести стандарт, общепризнанный в стране и за рубежом. Первый процесс неумолимо ставит перед администрацией вопрос о недалекой (уже через три-четыре года!) необходимости начать обновление парка школьных ЭВМ. Второй приводит прогрессивно мыслящих учителей к необходимости искать возможность приобрести современные ПЭВМ, совместимые с компьютерами фирмы IBM. В некоторых регионах это делают централизованно и широкомасштабно (например, в Москве, см: ИНФО 1991, № 5). Другие, не имеющие соответствующих средств для закупки IBM PC, стремятся разными путями получить в школу хотя бы одну такую машину. И этих примеров становится все больше.

Рассмотрение всей совокупности вопросов, связанных с настоящим периодом, проведем на примере московской СШ № 231, на базе которой создан научно-методический центр «Агат» (НМЦ «Агат»). В статье предлагается вариант общей стратегии использования ЭВМ в школе в этот переходный период, выработанный и апробированный сотрудниками центра. Для обоснования реальности предлагаемой стратегии — немного из истории компьютеризации нашей школы.

Первые школьные ЭВМ «Агат-7» мы получили в порядке централизованных поставок в 1988 г. Сразу поставили перед собой задачу-максимум — превратить школу в экспериментальную площадку новых информационных

технологий образования. Через год еще получили комплекты «Агат-7», «Агат-9» и два «Amstrad», совместимых с IBM PC/XT. За это время сформировался коллектив молодых учителей информатики, математики, физики, объединившихся в НМЦ «Агат». Все учителя нашего творческого коллектива помимо занятий с детьми на уроках, в кружках и на факультативах много работают над созданием нового программного и аппаратного обеспечения для школьных ЭВМ. Среди работ, которые мы выполнили самостоятельно или в сотрудничестве с другими авторами, можно назвать описанные в «ИНФО» пакеты программ для «Агата-7» и «Агата-9»:

редактор национальных текстов «Диалект»;

графический редактор «МаркиС» с мышью;

«Информатика-90»;

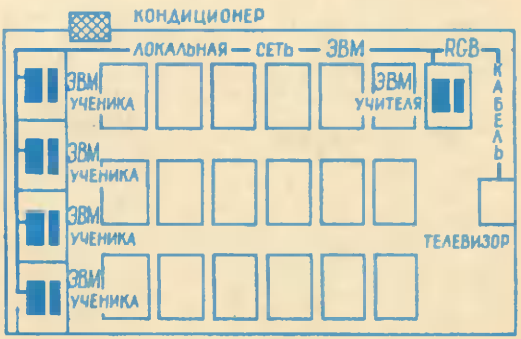
джойстик для «Агата-7, 9» и пакет игр для него;

«Раскрашка» для начальной школы и детсадов и др.

Научный центр программных средств обучения (НЦПСО) финансировал многие из вышеназванных разработок. С осени 1991 г. наша школа является базовой школой НЦПСО в «Московском проекте» информатизации образования (см.: ИНФО. 1991 № 5), в ней установлены два комплекта IBM PC/AT. Вполне резонно встал вопрос об использовании еще не выработавших свой ресурс «Агатов». Что же мы предлагаем? Как выглядит наша схема экспансии новых информационных технологий в общеобразовательные предметы средней школы?

Если компьютеров в школе мало, они, несомненно, остаются в КВТ. Но здесь — непременное условие: все «Агаты» объединяются в локальную сеть (разработка НЦПСО — НИИ ВК, серебряная медаль ВДНХ). Учительский «Агат» соединяется через последовательный порт RS-232 по протоколу Kermit-69 с одной IBM PC XT/AT, которая используется в трех назначениях. Во-первых, хранение данных и программ с «Агата», во-вторых, возможно использование «Агата» для подготовки данных (текстов, рисунков и т. д.) с последующей их обработкой на IBM — ведь весь класс за одну PC не посадишь! (Возможно и обратное — например, готовые рисунки с IBM использовать для разработки программ на «Агате». Графика здесь, конечно, беднее, но качество рисунков остается вполне высоким, что доказал опыт их использования в наших последних программах для «Агата» (см. слайды на обложке). В-третьих, наконец сам учитель и наиболее способные дети будут иметь хорошую возможность подготовиться к будущей работе на IBM-совместимой технике. Уверены, что рано или поздно компьютеры такого уровня будут во всех школах.

Если школа уже приобрела комплект PC XT/AT, оставлять в КВТ «Агаты» нецелесообразно. Мы предлагаем переместить их в кабинеты математики, физики, черчения, в группы продленного дня. Конечно, это потребует дополнительных усилий по организации электроснабжения, охраны, разработке программного обеспечения и переподготовке школьных учителей-предметников. Как решить эти непростые задачи? В классе устанавливаются 4—6 компьютеров. Один на учительском месте, подключен к большому демонстрационному монитору-телевизору, расположенному в центре над классной доской. Телевизор должен работать в режиме приема телепередач, и в режиме монитора (в идеале с видеомагнитофоном). Учительский компьютер по локальной сети связан с 4—5 ученическими ЭВМ, компактно расположенными в особой зоне класса у задней стены (рис. 1).



Класс дополнительно оборудуется одним кондиционером, установленным в окне, ближайшем к ученическим компьютерам в дальней части класса. Большее количество кондиционеров нецелесообразно, так как их мощность на охлаждение превысит мощность выделяемого компьютерами и другими источниками тепла. Эти кабинеты желательно отремонтировать, создав новый интерьер, что важно для психологической подготовки учителей и учащихся к новому стилю работы. Варианты могут быть самые разные, все зависит от возможностей школы и ее спонсоров. Компьютеризированные кабинеты необходимо оборудовать пожарной и охранной сигнализацией, а для уменьшения расходов на этот вид работы имеет смысл планировать переоснащение сразу всего школьного этажа с общим контуром сигнализации (в перспективе и других этажей школы). Стандартного количества «Агатов» из КУВТ хватит только для трех кабинетов, оборудуемых по этой схеме, но переоборудование целиком этажа оправданно, так как неуклонное пополнение школьного парка ЭВМ — a priory доказанная тенденция в народном образовании.

Что полезного принесет школе переоборудование кабинетов? Перечислим определенный круг задач с указанием конкретных программных средств, уже готовых или разрабатывающихся для «Агата».

1. **Контроль знаний.** Спорить о том, может ли компьютер заменить учителя, мы не будем. Но простейший вариант контроля — по карточкам с выбором ответа из готового перечня — реализован в экзаменаторах «Простачок» и «Агат-Экзаменатор» учителями из г. Чебоксар. В любом случае краткий письменный опрос в начале урока для 5—6 учеников проводят многие учителя. И для них не будет психологически трудным перейти на компьютерную форму опроса, тем более что проверку ответов производит сама программа по схеме, определенной учителем заранее, что позволяет освободить учителя от рутинной проверки работ этого вида. Другие, более сложные программы контроля знаний на «Агате», удовлетворяющие требованиям компьютерного урока, авторам неизвестны.

2. **Программы-тренажеры.** Полезность этого вида работы с компьютером не вызывает сомнений. Очень многие темы школьного курса требуют простого запоминания, например единицы измерений и размерности физических величин, правописание предлогов и суффиксов, фразеологизмы иностранных языков, преобразования математических выражений и т. п. Имеются лишь отдельные примеры таких программ для «Агата». Здесь можно упомянуть в основном известные программы по химии, русскому языку и др., написанные на учебно-производственном языке Рапира. Их качество не во всем удовлетворительно, но на безрыбье и рак — рыба. В этом виде задач, решаемых на уроке с помощью компьютера, есть огромный пробел, который пытается заполнить наш научно-методический центр «Агат». Среди работ, начатых осенью 1991 г., можно назвать программы-тренажеры «Предлоги английского языка» (и аналоги по французскому, русскому языкам; возможна быстрая адаптация и к другим языкам — немецкому и др.), «Графики функций» и «Афинные преобразования» по математике. Уже готова «Раскрашка» для начальной школы. Интерфейс этой программы разработан специально для дошкольников и ориентирован на использование «мыши». Управляя с помощью «мыши» курсором на экране, можно выбрать один из 15 цветов «Агата» (на 8-цветных мониторах картинка не испортится) и закрасить один из предлагаемых контурных рисунков. Все рисунки выполнены очень качественно, частично на нашем графическом редакторе «МаркиС», частично получены на IBM PC и с помощью платы RS-232 и программы «Kermit-69» «перекачаны» на «Агат». Количество предлагаемых рисунков составляет несколько десятков и может быть по заказу увеличено. Планируем мы и другие работы в этом направлении. Минимальный набор программ-тренажеров по различным предметам должен быть готов к сентябрю 1992 г.

3. **Демонстрационные и моделирую-**

щие программы. Эти программы могут органично дополнить традиционный набор технических средств обучения на уроке, не заменяя, а дополняя обычные кодоскопы, диапроектор, телевизор. Программы этого вида могут оживить схему (компьютерный плакат), продемонстрировать медленно идущий, невидимый или опасный эксперимент (радиоактивность, биология под микроскопом и др.). Ситуация в этом виде программного обеспечения для школы тоже плачевная (по крайней мере, для «Агата»). Нами начаты работы в этом направлении, первые результаты которых можно ожидать только к осени 1992 г. Зато готовы к использованию аппаратные средства — подключение к учительскому «Агату» стандартного отечественного или импортного телевизора в режиме демонстрационного монитора при сохранении всех обычных режимов его работы.

4. **Компьютерные учебники и базы знаний.** Это самый сложный для разработки и неоднозначный для использования вид программ. С некоторыми оговорками его можно предложить для повышения интереса учащихся к предмету, ведь дети всегда тянутся к новому и необычному, тем более к компьютеру. Информация предоставляемая ученику такой программой, должна быть краткой, не перегружающей экран, по возможности динамичной и грамотно оформленной методистом, художником и дизайнером. Надо иметь возможность вернуться в любое место компьютерного учебника и перескочить на несколько шагов вперед. Необходима также возможность получения более подробной информации сверх минимума, определенного учителем. Даже такой неполный перечень требований к компьютерному учебнику говорит о сложности разработки программ этого вида, касающихся в том числе и ограничений со стороны аппаратной части ваших ЭВМ.

Для хотя бы частичного преодоления вакуума в этой области software нами разработан пакет программ «Инструмент», позволяющий конструировать простейшие Бейсик-программы типа «компьютерный учебник» в терминах конечного результата, т. е. пользователь получает пакет различных редакторов текстовой и графической страниц «Агата», с их помощью рисует картинки (функции: точка, линия, окружность, прямоугольник, заливка, смена цвета и другие, отсутствующие в стандартном «Агате-Бейсике»), создает текстовые кадры, а потом «Инструмент» сам генерирует выходную Бейсик-программу. Полученная программа может работать независимо от «Инструмента» в любой Бейсик-среде, ее можно дополнить более сложными Бейсик-операторами, если вы знакомы с тонкостями Бейсика. Таким образом, «Инструмент» освободит вас от рутинной работы по манипулированию операторами PLOT, PRINT, COLOR, RIBBON, HTAB, VTAB и им подобными при разработке компьютерных учебников.

Наши идеи построены на том, что все предлагаемое и вновь разрабатываемое программное обеспечение имеет аналоги на машинах типа IBM PC XT/AT. Только при этом условии возможен безболезненный переход к новому стандарту школьных ЭВМ, который неизбежно рано или поздно состоится. Все работы, которые ведет наш коллектив, построены на этом принципе. Например, текстовый редактор «Диалект» с оконным интерфейсом и библиотекой изменяемых специальных и национальных шрифтов имеет в качестве прототипа известный редактор для IBM «Лексикон». А графический редактор «МаркиС» написан по образу и подобию «DOCTOR HALO» и также ориентирован на «мышиный» интерфейс. Еще один исповедуемый нами принцип — ориентация на максимально широкое использование манипуляторов «Джойстик-Агат» и «Мышь-Агат» в управлении программами. Это особенно важно, когда компьютер попадает в руки слабо подготовленного (в компьютерном смысле) учителя, который может испытывать физиологический страх перед обилием кнопок на клавиатуре и сложностью управлением ЭВМ.

НМЦ «Агат» привлек к разработке сценариев компьютерных учебников большой коллектив учителей, методистов, художников и программистов и надеется на успешную работу в этой об-

ласти.

Подведем итоги. Изложенные мысли являются не просто мечтами. Это стратегический план работы научно-методического центра «Агат» и школы № 231 Москвы на ближайшие 2—3 года. Учитывая, что школа наша самая обыкновенная (в чем можно убедиться лично — приглашаем всех желающих в гости для сотрудничества и обмена опытом), мы считаем эти планы реальными и для других школ, независимо от типа ЭВМ, которым они оснащены. В качестве доказательства мы можем предложить к продаже пакеты программ указанных типов для IBM PC XT/AT, КУВТ-86, УКНЦ, «Ямаха MSX-1,2» и, естественно, для «Агата-7, 9».

Наш адрес: 127540, Москва, ул. Дубнинская, д. 23, СШ № 231, кабинет информатики, НМЦ «Агат»; телефоны: 489-00-91, 487-86-68, 272-26-71.

Н. БИЛАК, С. РАКОВ

Учебная экспертная система ЭСОР

Одной из основных задач курса ОИВТ в средней школе и педагогическом вузе является ознакомление с различными деловыми применениями компьютеров, к которым традиционно относят работу с текстовыми и графическими редакторами, электронными таблицами, базами данных. Вместе с тем одним из наиболее перспективных деловых применений компьютеров являются экспертные системы (ЭС).

Для ответа на вопрос «Что такое ЭС?» воспользуемся «функциональным» подходом: ЭС — это диалоговая компьютерная система, оперирующая базой знаний в некоторой предметной области и с помощью встроенных средств логического вывода обеспечивающая ответы на запросы пользователя, касающиеся предметной области. ЭС должна также предъявлять обоснование ответа по запросу пользователя.

В связи со сказанным выше представляется целесообразным использование ЭС как в учебном процессе (в качестве инструмента), так и с целью ознакомления с идеями их построения и функционирования. При этом на первый план следует ставить концептуальную ясность учебных ЭС, их доступность, удобство в работе, надежность, а не производительность.

Именно с такой целью была разработана ЭСОР (аббревиатура от «Экспертная система типа объекты — реакции»). Первоначально ЭСОР предназначалась для применения в области химии, но затем была несколько изменена с целью выделения функций экспертной оболочки («пустой» экспертной системы) для расширения области применения (например, диагностика неисправностей или заболеваний).

Для наглядности будем рассматривать ЭСОР в контексте применения в химии, а для разъяснения возможностей ЭСОР в качестве экспертной оболочки приведем математическую модель, лежащую в ее основе. Акцент делается на практическом использовании ЭСОР, а вопросы ее построения и функционирования детально не обсуждаются.

Назначение. ЭСОР предназначена для отыскания цепочки реакций, которые необходимо выполнить для получения нужного вещества из заданного набора веществ посредством реакций из заданного набора реакций, обеспечивает диалоговые средства ведения базы знаний, состоящей из наборов веществ и реакций, и выбора интересующих баз данных объектов и реакций.

ЭСОР дает ответы на запросы следующих типов (в контексте выбранных ранее баз данных):

установить возможность получения заданного вещества;

найти множество всех веществ, которые можно получить;

найти способ получения данного вещества;

найти все способы получения данного вещества.

Математическая модель. Пусть X — множество допустимых объектов, $M(X)$ — множество всех подмножеств X, R — частичное отображение на $M(X)$, $R : M(X) \to M(X)$, $A \subset X$ — заданное множество объектов. Обозначим через $T(A) = \cup \{R(B) : B \subseteq A\}$ множе-

ство объектов, получаемых из подмножеств множества A посредством частичного отображения R.

Положим:

$H_1 = A$

$H_2 = T(A) \cup H_1$

$H_3 = T(H_2) \cup H_2$

.....

$S(A) = \cup \{H_i : i \in N\}$

Способом получения элемента $x \in S(A)$ назовем упорядоченное множество пар $\{\langle B_i, R(B_i)\rangle\}_{i<n}$ удовлетворяющее условиям:

$B_1 \subseteq A$;

$(\forall i > 1)\ B_i \subseteq A \cup (\cup \{R(B_j) : j < i\})$;

$(\forall i < n)\ x \notin R(B_i)$;

$x \in R(B_n)$.

Запросы (для заданного A):

для произвольного $x \in X$ установить его принадлежность множеству S(A);

описать множество S(A);

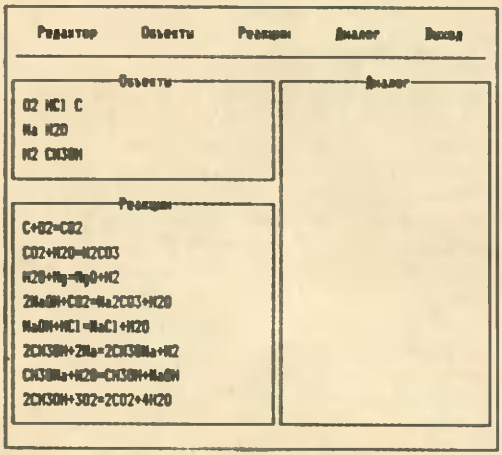
для произвольного $x \in S(A)$ описать способ получения x из A (произвольный);

все способы получения x из A.

Работа в системе. ЭСОР представляет собой многооконную систему, работа с которой поддерживается вложенной системой меню.

Войдя в режим редактирования, пользователь может заполнять базы данных «Объекты» и «Реакции», используя общепринятые представления для веществ и реакций.

На рис. изображены исходное меню и окна системы ЭСОР.



Загрузить готовые файлы данных можно из главного меню системы, выбрав режим «Объекты» или «Реакции». В этом случае пользователь попадет в подменю, предлагающее продолжить уточнение выбора режима работы:

загрузить — выдаются имена файлов текущего директория, которые можно загрузить в соответствующее окно;

новый файл — очистка соответствующего окна для создания нового файла;

записать в — запрашивается имя и сохраняется текущая база данных в файле с указанным именем;

записать — сохраняет содержимое базы данных под старым именем.

Для пояснения работы с системой в режиме «Диалог» приведем пример протокола такой работы. Вопросам компьютера предшествует знак «>>», при этом используется база данных, которая была приведена на рис.

```
>> что нужно получить?
   Na2CO3
>> указывать способ получения?(д/н)
   д
>> Na2CO3
>> CH3OH    есть в базе данных
>> Na       есть в базе данных
>> реакция:
>> 2CH3OH+2Na=2CH3ONa+H2
>> H2O      есть в базе данных
>> реакция:
>> CH3ONa+H2O=CH3OH+NaOH
>> C        есть в базе данных
>> O2       есть в базе данных
>> реакция:
>> C+O2=CO2
>> реакция:
>> 2NaOH+CO2=Na2CO3+H2O
>> продолжать поиск способов?(д/н)
   д
>> Na2CO3
>> способов нет
```

Ответив «н» на вопрос «Указывать реакции? (д/н)», пользователь получит лишь сообщение о том, можно ли получить данное вещество, без указания цепочки реакций.

Еще один пример протокола диалога пользователя с системой:

```
>> что нужно получить?
   все
>> CO2
>> H2CO3
>> H2O
>> Na2CO3
>> NaCl
>> H2
>> CH3ONa
>> NaOH
>> CH3OH
```

Реализация. ЭСОР реализована в среде Турбо-Пролог 2.0 для IBM PC. Объем исполнимого модуля — 20 К. Большую часть программы составляют модули поддержки дружественного интерфейса и обеспечения интерактивного ведения баз данных «Объекты» и «Реакции».

Центральной частью программы является процедура, реализующая отыскание необходимой цепочки реакций. Отметим, что на внутреннем языке системы подготовленные пользователем данные представляются в виде

набора фактов object (‹вещество›) и reaction (‹левая_часть›, ‹правая_часть›). При этом левая_часть — список веществ, стоящих в реакции слева от знака равенства, правая_часть — стоящих справа от знака равенства.

```
search(X,_):-str_int(X,_),!.
search(X,N):-
  object([X]),nl,
  write(X," есть в базе данных"),nl,!;
  reaction(Z,Z1),
  keeping(X1),
  not(member(X1,Z)),
  procedure(Z,Z1,[X],N),!.
procedure(Z,Z1,X,N):-
  N1=n+1,
  entry(X,Z1),
  retract(reaction(Z,Z1)),
  searchList(Z,N1)
  write(" реакция: "),nl,
  writer(Z,Z1),nl,!,
  level(N),
  assert(reaction(Z,Z1)).
searchList([],_):-!.
searchList([X|Y],N):-
  N1=N+1,N2=N1+1,
  search(X,N1),
  searchList(Y,N2),!.
```

```
searchList(_,_):-
  clearwindow,!,
  fail.
level(1):-
  nl,
  write("продолжать поиск
                способов?(д/н)"),
  readln(L),!,
  response(L).
level(_).
response("н"):-!.
response("д"):-clearwindow,!,fail.
```

По нашему мнению, приведенный фрагмент программы является характерным для Пролог-систем, поэтому его можно рассматривать совместно с ЭСОР на учебных занятиях по экспертным системам, логическому программированию, программированию на Прологе.

Апробация. ЭСОР используется как объект изучения в спецкурсе «Логическое программирование» на физико-математическом факультете Харьковского педагогического института им. Г. С. Сковороды. Совместно с сотрудниками кафедры химии подготовлены базы данных по различным темам неорганической химии.

Е. УМРЮХИН, Т. РОДИОНОВА, В. ПРОСВИРКИН, Т. ДЖЕБРАИЛОВА, А. ФОМИН
НИИ нормальной физиологии им. П. К. Анохина АМП СССР

Индивидуальные психофизиологические особенности школьников при работе на компьютере

Перестройка школы сопровождается внедрением компьютеров в педагогический процесс и в специальную подготовку учащихся. Однако известно, что продолжительная работа на ЭВМ, особенно в диалоговом режиме на дисплее, оказывает глубокое воздействие на организм человека, что может приводить к нервно-эмоциональному перенапряжению, нарушению сна, ухудшению зрения [1—4]. Комплексное обследование школьников показало, что занятия основами информатики и вычислительной техники связаны со значительными умственными, зрительными и психоэмоциональными нагрузками [2, 3]. При этом неблагоприятное воздействие работы на компьютере на организм человека обнаруживалось лишь у части обследованных (от 15 до 70 % по данным разных авторов), [1—4].

Однако причины различий в характере и степени негативных проявлений у разных лиц остаются неисследованными.

Целью настоящей работы было выявление факторов, обусловливающих возникновение нервно-эмоциональной напряженности у школьников при работе с компьютерами на уроках информатики и вычислительной техники.

В исследовании приняли участие более 50 школьников десятых классов двух московских школ, не имеющих отклонений в состоянии здоровья. В одной из школ кабинет информатики оборудован компьютерами ДВК, в другой — «Агат». Гигиенические условия кабинетов соответствовали требованиям.

Для оценки нервно-эмоционального напряжения школьников мы использовали статистические характеристики сердечного ритма. Регистрировали ЭКГ перед началом и в процессе работы на компьютере, заключавшейся во вводе составленной школьником программы. У 20 школьников ЭКГ регистрировали в течение всего периода работы с

программой. Вычисляли среднее значение и коэффициент вариации /v/ R—R интервала ЭКГ; строили гистрограмму и определяли модальное значение, амплитуду моды (Ам) R—R интервала и индекс напряженности (ИН) по Баевскому [5]. Тестировали личностные качества с помощью опросника EPI [6], определяли уровень личностной тревожности по тесту Спилбергера [7]. Достоверность различия средних оценивали с помощью критерия Стьюдента. Все расчеты проводили на ЭВМ МИКРО-2000.

Значения ЭКГ-показателей и их динамика при переходе от спокойного состояния к работе с программой на компьютере имели индивидуальную вариативность. У 4-х человек низкая нервно-эмоциональная напряженность, наблюдавшаяся в спокойном состоянии, сохранялась и во время работы (1-я гр.). Для 19 школьников во время работы было характерно умеренное повышение исходно низкой напряженности (2-я гр.). У 17 человек (3-я гр.) с началом ввода программы длительность и вариативность R—R интервала ЭКГ снизились, а Ам и ИН возросли до значений, достоверно ($p<0,001$) отличающихся от наблюдавшихся в 1-й и 2-й группах, свидетельствуя, таким образом, о высокой нервно-эмоциональной напряженности. У 7 школьников высокая напряженность механизмов регуляции сердечной деятельности отмечалась как в состоянии относительного покоя, так и во время работы (4-я гр.). И наконец, для 3 человек было отмечено увеличение длительности и вариативности R—R интервала и уменьшение ИН во время работы по сравнению с фоном (5-я группа).

В целом у 70 % школьников с началом работы на компьютере наблюдались изменения статистических характеристик сердечного ритма (сокращение среднего и модального значений R—R интервала ЭКГ и его вариативности и возрастание Ам и ИН различной степени выраженности), свидетельствующие о нарастании нервно-эмоциональной напряженности.

Результаты анализа ЭКГ были сопоставлены с данными психологического обследования. Оказалось, что школьники, вошедшие в 3-ю группу, характеризующуюся высокой нервно-эмоциональной напряженностью при работе с компьютером, достоверно отличаются от всех остальных минимальными значениями результатов тестирования по шкале интроверсия — экстраверсия ($p<0,01$). Индивидуальный анализ показал, что 13 человек из 17 были ярко выраженными интровертами и только 1 оказался экстравертом. Напротив, из 23 человек, вошедших в группы с низкой напряженностью (1-я) и с умеренным ее повышением (2-я) во время работы, 13 являлись экстравертами и только 4 — интровертами.

По шкале нейротизма и степени личностной тревожности различий между школьниками, составившими разные группы, выявлено не было.

В целом полученные данные свидетельствуют о достоверно большей нервно-эмоциональной напряженности при работе на компьютере у школьников-интровертов по сравнению с экстравертами. Этот факт может быть объяснен с учетом предложенной Г. Айзенком физиологической модели экстраверсии — интроверсии [8, 9]. Экстраверты, согласно Айзенку, характеризуются низким уровнем возбуждения ретикулярной формации головного мозга и, соответственно, необходимостью стимуляции из внешней среды для достижения оптимального уровня активации. Этим обусловлена направленность их активности на внешний мир, стремление к новым впечатлениям, общительность. Напротив, интроверты характеризуются высоким уровнем активации ретикулярной формации и им не требуется внешняя стимуляция. Это определяет направленность активности на внутренний мир человека, отсутствие потребности в новых впечатлениях, замкнутость.

Работа с компьютером, требующая постоянного диалога и обусловливающая дополнительные внешние стимулы, является, вероятно, более адекватной для школьников с преобладанием экстраверсии.

Анализ структуры деятельности школьников позволил вычленить некоторые устойчивые этапы: ввод программы, состоящей обычно из 10—12 операторов; результат работы программы (как правило, в первом случае ошибочный); один или несколько этапов исправления ошибок; достижение правильного результата. Кроме этих основных и обязательных этапов, работа над программой может включать консультации с преподавателем, обращение к учебным пособиям и другие этапы.

Наибольшее нервно-эмоциональное напряжение было зарегистрировано у всех без исключения школьников при получении на экране дисплея неправильного изображения. ЭКГ-характеристики при первом исправлении значимо не отличались от зарегистрированных при вводе программы. Дальнейшая динамика определялась результативностью деятельности. При многократных безуспешных попытках исправить ошибку в программе графического изображения отмечались изменения характера гистрограмм R—R интервала ЭКГ, свидетельствующие о нарастании напряженности механизмов регу-

ляции сердечной деятельности (например, при пятой попытке исправления программы построения графического изображения по сравнению с этапом ввода V снизился с 8,08 до 3,48 %; Ам увеличилась с 15 до 30 %, а ИН — с 28 до 169 ед.). Соответственно и после окончания урока характеристики сердечного ритма оставались измененными. В других случаях, при аналогичной начальной динамике характеристик ритма при получении правильного изображения все показатели в течение 1,5 мин достигали значений, близких к фоновым.

Специфическим фактором, обусловливающим резкое повышение нервно-эмоциональной напряженности, является сбой в работе компьютера. Следует отметить, что в литературе приводятся данные о возникновении раздражительности у операторов ЭВМ при отказах систем.

Нам удалось зарегистрировать объективную картину изменения ЭКГ у школьников при внезапных сбоях в работе компьютера. Отказ в работе компьютера и самостоятельная попытка его устранения приводят к достоверному ($p<0,01$) уменьшению длительности (с 678±7 мс до 629±±5 мс) и вариативности (с 7,65 до 4,58 %) R—R интервала ЭКГ. После устранения помех с помощью преподавателя самостоятельная работа с программой проходила на фоне минимальной вариативности R—R интервала (4,09 %). При этом длительность R—R интервала увеличивалась до значений, соответствующих фоновым (705±5 мс). В интегральном виде эти изменения отразились в возрастании ИН (до 180 ед. по сравнению с 73 ед. при работе до сбоя). Таким образом, сбой в работе компьютера привел не только к непосредственному увеличению напряженности, но и к изменению соотношения длительности и вариативности R—R интервала при возобновлении деятельности.

В целом наши исследования показали, что во время работы на компьютере у 70 % школьников наблюдались изменения статистических характеристик сердечного ритма (сокращение среднего и модального значений R—R интервала ЭКГ и его вариативности и возрастание индекса напряженности) различной степени выраженности. При этом степень нервно-эмоциональной напряженности при работе на компьютере достоверно выше у школьников-интровертов по сравнению с экстравертами.

Достижение полезного результата является фактором, снимающим возникшее у учащихся во время работы на компьютере нервно-эмоциональное напряжение, и, напротив, неэффективность действий школьника приводит к его возрастанию.

Приведенные выше результаты психофизиологического исследования учащихся в процессе их работы на компьютере свидетельствуют о дифференциации школьников по типу их биологической реакции при диалоговом общении с ЭВМ, которая в значительной мере определяется их психологическим статусом, в частности уровнем интро- и экстраверсии. Поскольку для интровертов диалог с компьютером объективно более сложен, чем для экстравертов, то это следует учитывать при обучении данной группы школьников основам информатики. Конкретно можно порекомендовать для данной группы учащихся более углубленное теоретическое изучение программирования, прежде чем они приступят к практической реализации знаний на ЭВМ. Это сократит диалоговый аспект работы, сделав его более эффективным. Такая тактика обучения школьников с выраженной интроверсией позволит повысить эффективность обучения на уроках информатики и избежать нежелательных психофизиологических реакций. Для экстравертов, напротив, может быть рекомендовано преимущественное обучение программированию в диалоговом режиме, что соответствует их психологическому типу и способу психофизиологического реагирования. Поскольку наши исследования показали, что по успеваемости школьники интро- и экстраверты не различаются, выполнение этих рекомендаций обеспечит снижение «физиологической стоимости» процесса освоения школьниками работы на компьютере.

Совершенно необходимым требованием с позиций профилактики возможных негативных влияний на состояние здоровья является контроль за обязательным достижением школьником правильного результата при работе на компьютере.

Особое значение в плане профилактики играет надежность оборудования кабинета информатики, так как отказы в работе компьютера приводят к резкому возрастанию нервно-эмоционального напряжения у школьников.

Литература

1. Видеодисплейные терминалы и здоровье пользователей. ВОЗ, Женева, 1989. 150 с.

2. *Гуменер П. И., Комарек Л., Провазник К., Синякина А. Д.* Функциональные возможности школьников 10—14 лет при работе на микрокомпьютере // Новые исследования по возрастной физиологии. 1987. № 1 (28). С. 69—72.

3. *Леонова Л. А., Савватеева В. С., Бирюкович А. А.* и др. Физиолого-гигиеническая оценка функционального состояния школьников при изучении ими «Основ информатики и вычисли-

тельной техники» // Новые исследования по возрастной физиологии. 1987. № 2 (29). С. 70—72.

4. *Леонова Л. А.* Физиолого-гигиенические основы организации учебных занятий школьников с применением электронно-вычислительной техники // Новые исследования в психологии и возрастной физиологии. 1989. № 1. С. 78—81.

5. *Баевский Р. М.* Математический анализ изменений сердечного ритма при стрессе. М.: Медицина, 1984. 221 с.

6. Eysenck H. Y., Eysenck S. B. Manual for Eysenck personality inventory. L., 1964.

7. *Ханин Ю. П.* Краткое руководство к применению шкалы реактивной и личностной тревожности И. Д. Спилбергера. Л., 1976. 15 с.

8. Eysenck H. Y. The bioloqical basis of personality. Springfield, 1967.

9. *Русалов В. М.* Новый вариант адаптации личностного теста EPI // Психологический журнал. 1987. Т. 8. № 1. С. 113—126.

К нашим читателям

Множество отдельных программ и пакетов предлагаются самыми разными продавцами — от учколлекторов до частных лиц. Есть продукция профессионалов и любителей, на Бейсике и в кодах, для домашних компьютеров и школьных, для уроков информатики, родного и иностранных языков, физики, химии, географии... В общем, чрезвычайное разнообразие программ — и в то же время крайняя нехватка информации о них.

Чтобы помочь вам ориентироваться в этом разнообразии, «ИНФО» открывает soft-парад. Его цель — служить дополнительным источником информации об ассортименте и качестве программ, исходящей, что очень важно, от ваших коллег-практиков. На первых порах главной задачей будет не оценивать, а информировать об имеющихся программах, чтобы пользователи знали, что и из чего можно выбирать.

Это предприятие будет успешным только при условии активного участия большого числа людей, иначе говоря, успех и полезность soft-парада зависят от вас. Пишите! И сообщайте о тех программах, с которыми вам приходилось работать.

Желательно, чтобы в письмах были по возможности полно приведены следующие сведения:

название программы;

ее автор;

длина файла;

для какого компьютера предназначена;

где получена, на каких условиях (куплена, обменена, создана самостоятельно...);

назначение (обучающие, контролирующие, моделирующие — по каким предметам; системные — утилита, драйвер и т. д.; прикладные — текстовый редактор, база данных и т. д.; инструментальные — транслятор, система для создания обучающих программ и т. д.); краткая характеристика (несколько строк) в произвольной форме;

ваша оценка (—2 — очень плохо; —1; 0, +1; +2 — очень хорошо) качеств программы (удобство интерфейса, простота использования, эффективность применения, увлекательность, общая оценка);

основные сведения о себе (возраст, профессия, стаж, пользовательский стаж, как и для чего используете компьютер — учите или учитесь с его помощью в школе, играете дома, работаете дома и т. д.).

Надеемся также получить ваши предложения по организации парада: как классифицировать программы, какие параметры оценивать; быть может, стоит оценивать и компьютеры; а может быть, и продавцов программ...

Ждем ваших писем!

Хотел бы обмениваться программами для БК-0010.01.

236006, Калининград, ул. Октябрьская, 27, кв. 49. Ю. Б. Чернай

Хотел бы переписываться и обмениваться программами с владельцами БК-0010.01. Банк программ — около 800. Программирую на ассемблере.

352100, Краснодарский край, г. Тихорецк, ул. Гоголя, 75, кв. 25. А. А. Кучинский

Хочу обмениваться программами и информацией с владельцами «Корвета». Имею более 500 игровых, системных и обучающих программ.

334030, Крымская АССР, Джанкойский р-н, с. Кондратьево, ул. Ленина, 1, кв. 21. А. В. Чижиков

Наш компьютерный клуб «РИС» хочет завязать переписку с пользователями ПЭВМ «Ямаха» MSX-2. Имеем более 80 игровых (фирменных), 50 обучающих, 20 оригинальных программ.

741405, Узбекистан, Хорезмская обл., г. Хива, газгородок, д. 2, кв. 10; тел.: 5-75-51, 5-51-30, 5-70-03. Рустам Сабиров

Предлагаю обмен программами для «Спектрума».

668541, Республика Тува, Тоджинский р-н, п. Ырбан, ул. Зеленая, 9. И. Н. Родионов

Приглашаю к обмену программами для «Вектора-06Ц».

652810, Кемеровская обл., г. Осинники-11, а/я 63. Н. И. Демин

Хотел бы переписываться и обмениваться программами для БК. Имею более 600 программ.

380061, Тбилиси, Глданский массив, м/р «А», корп. 10, кв. 82. О. В. Мартынов

Приглашаю к обмену игровыми и другими программами пользователей МК-85, 52, 61.

117465, Москва, ул. Тюленева, 41—108. Тел. 337-08-17. М. Т. Лялин

Хочу переписываться и обмениваться программами для УКНЦ, «Спектрума», IBM. Готов ответить любому написавшему.

333000, Симферополь, ул. Карла Маркса, 4, кв. 17. Д. В. Ревенко

Имею довольно много программ для БК-0010. Предлагаю переписку, обмен программами, опытом, информацией. Немедленно вышлю полный каталог.

202020, Эстония, Кохтла-Ярве, ул. Катсе, 11, кв. 48. Д. В. Поспелов

Предлагаю объединиться тем, кто хочет заниматься (или уже занимается) информатикой и ВТ, но не имеет доступа к ЭВМ.

474321, Целиноградская обл., Вишневский р-н, с-з Ижевский, ул. Советская, 9. В. Шмидт

Предлагаю обмен программами для ПК «Партнер-01.01», хотел бы переписываться с владельцами этого компьютера.

680035, Хабаровск-35, а/я 1718. М. М. Лейбович

Хотел бы переписываться с владельцами ПК «Вектор-06Ц». Имею также «Спектрум» и товарищей, владельцев других моделей ПК. Отвечу всем без исключения.

342000, Донецкая обл., г. Константиновка, а/я 34; тел. 2-65-92. Д. В. Литовка

Буду переписываться и меняться программами для ПЭВМ «Апогей-БК01», РК-86 и совместимых с ними.

624300, Свердловская обл., г. Кушва, ул. Коммуны, 82а, кв. 98. С. Ю. Боровиков

Предлагаю для обмена БК-0011 на принтер с интерфейсом ИРПР, шириной печати 120—180 знаков и БК-0011 на дисковод «Электроника МС 5309».

643050, Северо-Казахстанская обл., Пресновский р-н, с. Пресновка, ул. Труда, 4, кв. 22. Тел. 2-13-81. Г. Валь

Занимаюсь обменом и тиражированием программ для компьютера, совместимого с «ZX-Спектрум». В письмо вложите конверт с вашим адресом.

663080, Красноярский край, Дивногорск, ул. Заводская, а/я 413

С. АХМАНОВ, А. НЕЧАЕВ, Н. РОЙ, А. СКУРИХИН

Архитектура «Корвета»

Чтобы лучше понять, как, обращаясь к памяти и управляющим регистрам, получить необходимые надписи и эффекты на экране, желательно познакомиться с устройством дисплея. Приводимая ниже техническая информация, на первый взгляд не нужная программисту, пригодится на определенном этапе совершенствования и ему.

В качестве устройства отображения информации в «Корвете» используется электронно-лучевая трубка (ЭЛТ). Принцип формирования изображения аналогичен используемому в телевидении. Электронный луч, скользя по экрану, оставляет на нем светящийся след. Специальное устройство, называемое блоком разверток, заставляет луч вычерчивать строки: начиная с верхней левой части экрана, он проводит горизонтальную линию, затем возвращается к левому краю, но уже несколько ниже прежнего положения, и опять проводит горизонтальную линию. Таким образом луч пробегает по всей поверхности экрана и затем возвращается в исходное положение (верхний левый край экрана), формируя кадр.

Управляя интенсивностью электронного луча, можно разбивать строки на точки и рисовать таким образом любые фигуры. Это делает видеосигнал, формируемый контроллерами А/Ц и графического дисплеев. Отметим, что изображение формируется только при прямом ходе развертки — при движении луча слева направо. При обратном ходе, когда луч возвращается в начало следующей строки или в начало кадра, им управляет блок разверток с помощью сигналов *вертикальный и горизонтальный бланк*, которые отключают луч, так что следа на экране не остается.

Получаемое таким образом изображение называют растровым.

Длительность полной строки (включая обратный ход) составляет 64 мкс и соответствует телевизионному стандарту. В ранних версиях «Корвета» иногда использовалась длительность 65,6 мкс (отход от стандарта был вызван плохим качеством мониторов). Длительность прямого хода горизонтальной развертки (т. е. время, в течение которого формируется изображение строки) — 51,2 мкс. Если его сопоставить с разрешением по горизонтали (512 точек), становится ясно: чтобы отобразить одну точку, надо включить луч на 100 мс.

Число отображаемых строк соответствует разрешению по вертикали и равно 256, таким образом длительность прямого хода вертикальной развертки составляет $256 \times 64 = 16{,}384$ мс. Частота смены кадров — 50,08 Гц, что также близко к телевизионному стандарту. Отметим, что время обратного хода вертикальной развертки — 4096 мкс — довольно велико в масштабе процессора.

Теперь можно перейти к устройству контроллера А/Ц дисплея. Его функциональная схема приведена на рис. 1.

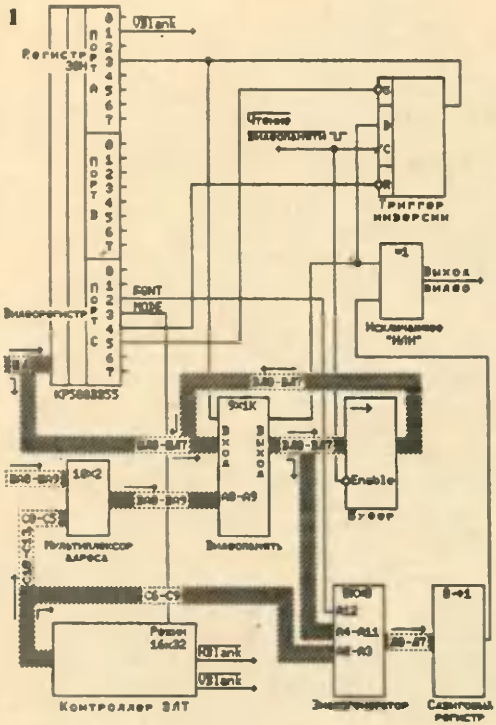
Известно, что экран А/Ц дисплея способен отобразить 16 текстовых строк (не путать с телевизионными строками), по 64 символа в строке, всего 1024 символа. Уже говорилось, что в состав А/Ц дисплея входят 10249-битовые ячейки. Каждой из них соответствует свое место на экране, и высвечивается на этом месте символ, код которого записан в соответствующую ячейку. В 8 битах можно закодировать любой из 256 символов, а экзотический 9-й бит используется для хранения признака инверсии (определяет, будет ли символ белым на черном фоне или черным на белом).

Следующий важный узел — контроллер ЭЛТ. Он генерирует синхронизирующие и управляющие сигналы, определяет движение электронного луча по экрану и синхронно задает адрес ячейки видеопамяти, соответствующей текущей позиции экрана.

Контроллер ЭЛТ сделан «на жесткой логике», программирования не требует и не допускает, начинает работать сразу после включения питания. Отчасти управлять им, однако, может сигнал MODE (см. рис. 1), меняющий режим отображения. Если его значение 1, то отображаются 32 символа в строке, если 0 — то 64. Управлять же сигналом можно через бит 3 видеорегистра

Продолжение. Начало см.: Информатика и образование. 1992. № 1.

1



Функциональная схема А/Ц дисплея «Корвета». Данные с системной шины данных ШД поступают на вход видеопамяти и микросхему КР580ВВ55. На вход видеопамяти поступает и сигнал инверсии с триггера инверсии. Мультиплексор адреса выдает на шину видеоадреса ВА адрес данных в видеопамяти, синтезированный из адресов, поступивших с системной шины адреса ША, или состояний счетчиков контроллера ЭЛТ (С0 — С5 определяют номер символа в строке, С10 — С13 — номер текстовой строки). С выхода видеопамяти видеоданные ВД поступают через буфер на системную шину данных (если это предписывает сигнал чтения видеопамяти) и знакогенератор. Знакогенератор из видеоданных и состояний счетчиков контроллера ЭЛТ (С6 — С9 определяют номер телевизионной строки в пределах одной текстовой строки) формирует сигналы управления яркостью луча ЭЛТ. Сдвиговый регистр превращает байт данных в последовательность поочередно передаваемых битов. Видеовыход — полноценный видеосигнал, способный управлять лучом черно-белого алфавитно-цифрового монитора. HBlank и VBlank — сигналы гашения луча ЭЛТ при обратном ходе по горизонтали и вертикали; VBlank поступает также на один из выходов порта А.

(порт С), но об этом чуть позже.

Следующий элемент функциональной схемы — знакогенератор, формирующий изображения символов на экране.

Экран разбит на так называемые знакоместа — прямоугольники размером 8 точек по горизонтали и 16 по вертикали. Символы А/Ц дисплея могут высвечиваться только в этих прямоугольниках. С каждым знакоместом связана своя ячейка видеопамяти; нулевая ячейка соответствует левому верхнему углу экрана, далее адреса возрастают слева направо и сверху вниз, как это показано на рис. 2. Когда электронный луч пробегает по какому-либо знакоместу, из соответствующей ячейки видеопамяти извлекается код символа, который должен быть отображен. С изображением символа код не имеет ничего общего, он задает адрес таблицы, описывающей изображение. Собственно, набор этих таблиц и есть знакогенератор. Это просто массив байтовых ячеек памяти, каждая из которых несет информацию об одной строке (телевизионной) одного символа, т. е. о 8 из составляющих его точек. После того как байт выбран, дальнейшее преобразование записанной в нем информации производит сдвиговый регистр; но как найти нужный байт?

2

	0	1	2	...	62	63
1	64	65	66	...	126	127
	128	129	130	...	254	255
	.	.	.	...	.	.
	960	961	962		1022	1023

Поскольку каждый символ состоит из 16 строк, таблица символа занимает 16 байтов. Чтобы идентифицировать один из них, достаточно 4-битового номера строки (нетрудно догадаться, что его формирует контроллер ЭЛТ). Таблица символа, как уже было сказано, адресуется 8-битным кодом символа, считанным из видеопамяти. Таким образом и получается 12-битный (старшие биты — код, младшие — номер строки) адрес нужного байта.

	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0
1	0	1	1	1	1	1	1	0
2	0	1	1	0	0	0	0	0
3	0	1	1	0	0	0	0	0
4	0	1	1	0	0	0	0	0
5	0	1	1	0	0	0	0	0
6	0	1	1	1	1	1	1	0
7	0	1	1	0	0	1	1	0
8	0	1	1	0	0	1	1	0
9	0	1	1	0	0	1	1	0
10	0	1	1	0	0	1	1	0
11	0	1	1	1	1	1	1	0
12	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0	3
0									
1		*	*	*	*	*	*		
2		*	*						
3		*	*						
4		*	*						
5		*	*						
6		*	*	*	*	*	*	*	
7		*	*				*	*	
8		*	*				*	*	
9		*	*				*	*	
10		*	*				*	*	
11		*	*	*	*	*	*		
12									
13									
14									
15									

«Строение» таблицы, описывающей изображение символа, продемонстрировано на рис. 3. В левой части рисунка каждая строка представляет байт таблицы в двоичном представлении, а в правой — отображение на экране. Обратите внимание, что первым отображается старший бит считанного из знакогенератора байта, соответственно на экране он расположен слева.

В «Корвете» в качестве знакогенератора используется ПЗУ. Программируется оно на заводе, так что пользователю доступен только фиксированный набор символов. Есть, однако, возможность несколько расширить его. Дело в том, что микросхема ПЗУ, используемая в качестве знакогенератора, имеет емкость 8 Кбайт, в то время как 256 символов занимают, как легко сообразить, 16×256=4 Кбайт. Следовательно, можно иметь два набора по 256 символов каждый. Правда, их нельзя использовать одновременно. Менять наборы можно, воздействуя на бит 2 видеорегистра. Состоянию 0 этого бита соответствует основной набор. На рис. 1 соответствующий сигнал обозначен FONT; он добавляет 13-й бит к адресу знакогенератора. К сожалению, не все могут воспользоваться вторым набором: в некоторых «Корветах» он не запрограммирован.

Продолжим рассмотрение схемы дисплея. Сигнал с выхода сдвигового регистра поступает на элемент «исключающее или» (XOR), позволяющий управлять инверсией. Если на второй его вход поступает 0 из 9-го бита видеопамяти, то мы видим прямое изображение символа, если 1, то мы видим символ в «негативном» изображении — черным на светлом фоне.

Приведенные сведения нужны в основном для лучшего понимания архитектуры А/Ц дисплея. Теперь же начинается самое интересное — рассказ о том, как процессор может выводить информацию на экран.

Из функциональной схемы видно, что процессор имеет возможность напрямую обмениваться данными с видеопамятью. Ее начальный адрес в адресном пространстве «Корвета» зависит от текущей конфигурации (существуют конфигурации, в которых она недоступна, например 5СН, используемая в некоторых версиях МикроДоса). «Вывод» символа на экран заключается в записи его кода в ячейку видеопамяти, связанную с нужной позицией на экране.

Процессор записывает и считывает информацию байтами, размер же ячейки видеопамяти 9 битов. Доступ к 9-му биту осуществляется через триггер инверсии (см. рис. 1); когда процессор производит запись в видеопамять, в 9-й бит записывается состояние этого триггера.

Работой триггера инверсии управляют биты 4 и 5 видеорегистра. Если бит 5 установлен, а бит 4 сброшен, то триггер инверсии ставится в безусловно сброшенное состояние и все, что записывает процессор, отображается в прямом виде. Если бит 5 сброшен, а бит 4 установлен, то символы отображаются в инверсном виде. Если установлены оба бита, то триггер работает в режиме памяти, т. е. сохраняет свое предыдущее состояние. Впрочем, оно все же может измениться — в результате чтения из видеопамяти: при этом в триггер заносится состояние 9-го бита читаемой ячейки. Используя этот режим, процессор может копировать 9-битовые ячейки байтовыми обращениями, что бывает необходимо, например, при скроллинге (сдвиг экрана вверх или вниз).

Ситуация, когда оба бита сброшены, является запрещенной. Конечно, если она все-таки возникнет, ничего не сломается. В данном случае запрет означает отсутствие смысла.

Узнать состояние триггера инверсии (и проанализировать благодаря этому состояние бита инверсии любой ячейки видеопамяти) процессор может через бит 3 регистра 38Н (порт А на рис. 1). Кстати, к биту 1 этого же регистра подведен сигнал обратного хода вертикальной развертки VBlank. Пользуясь им, можно, например, синхронизировать работу программы с кадровой разверткой или измерять время.

На этом закончим теорию и перейдем к практике. Упражнения помогут вам использовать полученные знания, но сначала несколько общих замечаний.

Программы-примеры написаны преимущественно на Паскале, точнее Экспресс-Паскале (см. статьи М. Ковтуна). Почему не на ассемблере, который обычно используется для программ, управляющих «железом»? Паскаль гораздо понятнее и выразительнее, чем ассемблер, разобраться в описанном им алгоритме значительно легче, а цель именно в этом. При необходимости Паскаль-программу легко трансформировать на любой язык, в том числе и на ассемблер. Последний будет использоваться лишь в тех случаях, когда представление программы на Паскале невозможно (например, по причине недостаточного быстродействия). Кроме того, иногда пример на Паскале будет продублирован примером на ассемблере — если существует прием, значительно повышающий эффективность программы.

Почему не Бейсик? И потому, что алгоритм, записанный на Паскале, понятнее, и потому, что Бейсик имеет больше ограничений при управлении аппаратурой. Во

многих случаях Паскаль-программу можно переписать на Бейсике. Если для такой трансформации есть ограничения, они будут оговорены. Сразу укажем первое, довольно общее: поскольку некоторые версии МикроДос имеют базовую конфигурацию памяти 5CH, в соответствующих версиях дискового Бейсика адреса регистров смещены, а видеопамять и вовсе отсутствует в адресном пространстве процессора. Поэтому прямая трансформация возможна только в Бейсик, зашитый в ПЗУ, и в дисковый Бейсик, пригодный для CP/M.

Итак, пример 1. Эта программа позволит вам просмотреть содержимое знакогенератора. Компиляцию лучше проводить в память (это ускорит работу); записывать исходные тексты на диск можно, но не обязательно.


```
{пример 1}
program CharSet;
const VideoRamBase = $FC00;        {начало видеопамяти в конфигурации 1CH}
var   i       :integer;
begin
        ClrScr;                    {очистка экрана}
        for i := 0 to 255 do       {перебор всех возможных кодов символов}
          mem[VideoRamBase+i+i] := i; {запись этих кодов в видеопамять}
end.
```

Запустите эту программу и объясните себе, почему возникла такая картинка. Обязательно во всем докопайтесь до сути.

Мы говорили, что знакогенератор содержит два набора символов. Следующий пример позволит убедиться в этом. Если второй набор в вашем компьютере отсутствует, вы увидите белый экран.

```
{пример 2}
program CharSets;
const VideoRegister = $FB3A; { адрес
                               видеорегистра }
      Font          =      4; { маска
                              бита фонта }

procedure CharSet;
const VideoRamBase = $FC00;
var   i       :integer;
begin
        ClrScr;
        for i := 0 to 255 do
          mem[VideoRamBase+i+i] := i;
end;

begin                                 { тело
                                      программы }
        CharSet;                      { рисуем
                                      все знаки }
        while ReadKey <> #27 do
              { если ESC, заканчиваем }
          mem[VideoRegister] :=
        { переключаем фонт, не трогая }
          mem[VideoRegister] xor Font;
       {остальные биты видеорегистра}
          mem[VideoRegister] :=
        { восстановим основной фонт }
          mem[VideoRegister] and
                          (Font xor $FF);
end.
```

Здесь производится запись в видеорегистр; скажем несколько слов о работе с ним и с регистром 38H.

Оба эти регистра являются портами микросхемы параллельного интерфейса KP580BB55, включенного в состав регистров РЕГ1 (возможны чтение и запись, см.: ИНФО. 1992. № 1) и имеющего смещение (адрес относительно начала области РЕГ1) 38H. Это означает, что в конфигурации 1CH, принятой в CP/M, базовый адрес этого интерфейса равен OFB38H. Этот адрес есть адрес порта A, настроенного на ввод в режиме 0; мы называем его регистром 38H. Функцию видеорегистра выполняет порт C, настроенный на вывод в режиме 0. В конфигурации 1CH он имеет адрес OFB3AH.

Если мы что-либо записываем в видеорегистр, соответствующие сигналы мгновенно появляются на выводах интерфейса, тем самым воздействуя на электрические цепи. При чтении из него читается та информация, которую мы записали. Так в любой момент можно узнать, в каком состоянии находятся те или иные сигналы. Последнее используется весьма активно.

Отметим, что порт C допускает побитовую установку через регистр управления. Мы этой возможностью пользоваться не будем, а желающих разобраться с этим режимом отсылаем к описанию микросхемы KP580BB55.

Регистр 38H выполняет функцию ввода ряда сигналов. Попытка что-либо записать в него не приводит ни к каким результатам. При чтении из него процессор получает «слепок» состояния выводов порта A интерфейса.

А теперь попробуем переключить режим отображения (16×64 и 16×32). Программа, делающая эту операцию, отличается от предыдущей лишь номером бита, на который надо воздействовать.

```
{пример 3}
program Modes;
const VideoRegister = $FB3A;
      Mode          =      8; { маска
                              бита режима }

procedure CharSet;
const VideoRamBase = $FC00;
var   i       :integer;
begin
```

```
        ClrScr;
        for i := 0 to 255 do
          mem[VideoRamBase+i+i] := i;
end;

begin
        CharSet;
        while ReadKey <> #27 do
          mem[VideoRegister] :=
      { переключаем режим, не трогая }
          mem[VideoRegister] xor Mode;
       {остальные биты видеорегистра }
          mem[VideoRegister] :=
{ восстановим основной режим (16X64) }
           mem[VideoRegister] and
                        (Mode xor $FF);
end.
```

Маленький фокус: изменим в программе одну цифру, и широкие символы режима 16×32 пропадут с экрана.

```
{пример 4}
program Modes;
const VideoRegister = $FB3A;
      Mode          =     8;

procedure CharSet;
const VideoRamBase = $FC01;
var   i       :integer;
begin
        ClrScr;
        for i := 0 to 255 do
          mem[VideoRamBase+i+i] := i;
end;

begin
        CharSet;
        while ReadKey <> #27 do
          mem[VideoRegister] :=
          mem[VideoRegister] xor Mode;
          mem[VideoRegister] :=
          mem[VideoRegister] and
                        (Mode xor $FF);
end.
```

Происходит это потому, что в режиме 16×32 могут использоваться, точнее высвечиваются, только четные адреса видеопамяти — ведь для широких символов есть только 512 знакомест и соответственно достаточно вдвое меньшего числа ячеек памяти.

Следующие два примера помогут научиться выводить текст в произвольное место экрана, они различаются способами задания позиции экрана. В примере 5 явно присутствуют позиция в строке и номер строки, пример 6 демонстрирует доступ к экрану непосредственно по адресу. Нажатие клавиши ESC прекращает работу программ.

```
{пример 5}
program OutXY;
const
      VideoRamBase  = $FC00;
      MaxX          =  63;
      MaxY          =  15;
      MaxCod        = 255;
var     x,        { для хранения
                      координаты X }
        y,        { для хранения
                      координаты Y }
        c: byte;  { для хранения
                      кода символа }

begin
  ClrScr;
  Randomize;      {инициализируем
            генератор случайных чисел}
  while ReadKey <> #27 do
    begin
      x := Random(MaxX);
      y := Random(MaxY);
      c := Random(MaxCod);
      mem[VideoRamBase+x+64*y] := c;
     {запись этих кодов в видеопамять}
    end;
end.

{пример 6}
program OutXY;
const
       VideoRamBase  = $FC00;
       MaxAddr       =  1023;
       MaxCod        =   255;
var      Adr: integer; { для хранения
                  относительного адреса }
         c: byte;

begin
  ClrScr;
  Randomize;
  while ReadKey <> #27 do
    begin
      Adr := Random(MaxAddr);
      c   := Random(MaxCod);
      mem[VideoRamBase+Adr] := c;
    end;
end.
```

Продолжение следует.

С. АХМАНОВ, А. НЕЧАЕВ, А. СКУРИХИН

Архитектура процессора КР580ВМ80А

В повседневной жизни прерывания (правда, называем мы их обычно иначе) возникают очень часто. Вы готовите уроки (или завтрак); звонит телефон. Если вы отвлеклись от разговора, а потом вернулись к прерванному занятию, то этим в точности смоделировали компьютерное прерывание, т. е. переход по внешнему сигналу от одного дела к другому с последующим возвратом к первоначальному.

Источником прерывания послужил звонок телефона. Благодаря этому устройству вы сразу узнаете, что кто-то хочет связаться с вами. Представьте, что звонка нет (или он неисправен). Тогда, чтобы узнать, «звонит» кто-нибудь или нет, придется каждую минуту поднимать телефонную трубку и слушать. Человек на такое вряд ли согласится, а вот компьютеру приходится. Несмотря на кажущуюся абсурдность, этот способ ожидания событий (он называется опросом, по-английски polling) используется довольно часто для организации общения процессора со сравнительно медленно работающими периферийными устройствами, например клавиатурой и принтером.

Недостатки опроса очевидны. Из-за малого КПД (сотни проверок «на всякий случай»!) он отнимает значительное время у основной программы, хуже того — усложняет алгоритм ее работы. Ведь чтобы не пропустить нужное событие, мы должны «натыкать» блоки опроса в разные части программы.

Так что при введении механизма прерываний, несмотря на кажущееся усложнение (приходится изучать и использовать новые, дополнительные средства), программа, как правило, значительно упрощается, ее характеристики улучшаются и (на первый взгляд это может показаться странным) она начинает работать надежнее.

Расскажем о прерываниях подробнее. Предположим, что дополнительно к исполнению основной программы (пусть это будет война с космическими негодяями) компьютер должен отображать в углу экрана текущее время. Эти две задачи (игра и отображение времени) не связаны между собой. Игра — непрерывный процесс; реализующая ее программа должна перемещать какие-то объекты по экрану, реагировать на клавиши и т. д. Программа, имитирующая часы, должна следить за временем и каждую минуту обновлять показания на экране. Ясно, что ей не обязательно работать постоянно. Достаточно, если она будет включаться каждую секунду для увеличения счетчика и, если настала пора, вывода новых цифр на экран.

Для организации этого процесса используем электронный таймер, отсчитывающий время и подающий раз в секунду сигнал, который и будет событием, включающим программу-часы.

Прерывание реализуется на аппаратном уровне. Чтобы вызвать его, нужно подать сигнал (он называется запросом прерывания) на специальный вывод процессора. После этого, если так называемый триггер разрешения прерывания установлен (прерывания разрешены), процессор начинает цикл прерывания.

Он состоит из следующих операций. Сначала вместо считывания очередной команды основной программы из специального устройства, называемого контроллером прерываний, считывается команда вызова подпрограммы обработки прерываний (содержимое регистра PC при этом не изменяется). Выполняя ее, процессор запоминает адрес очередной команды текущей программы (т. е. содержимое PC) и передает управление программе, которая должна выполняться в связи с прерыванием — в нашем случае программе часов. Эта программа должна сохранить где-нибудь значения тех регистров процессора, которые она собирается изменить; выполнить свою основную задачу (увеличить значение счетчика, вывести на экран цифры); восстановить старые значения регистров, которые она использовала. Ее последней командой должна быть команда выхода из подпрограммы; она восстановит значение PC и тем самым возвратит управление основной программе, которая продолжится, как ни в чем ни бывало.

Описание конструкции процессора почти окончено, осталось упомянуть о способах общения с УВВ. В «Корвете» этим устройствам отводится часть ОЗУ, через которую процессор и обменивается данными с внешним миром; можно действовать и иначе — КР580ВМ80 имеет специальное адресное пространство ввода-вывода (отделенное от обычного 64-килобайтного) размером в 256 байтов. Ячейки этого пространства называются портами — видимо, чтобы подчерк-

Продолжение. Начало см.: Информатика и образование. 1992. № 1.

нуть их роль «ворот» для связи, а также чтобы не путать с ячейками обычной памяти.

Команды КР580ВМ80

Разобравшись со структурой процессора, перейдем к описанию команд. В памяти они записываются, естественно, двоичными числами, но программисты практически всегда обозначают их буквенными аббревиатурами, так называемыми мнемониками. Мнемоники достаточно просто перевести в двоичные коды: «в жизни» переводом занимается транслятор-ассемблер, но можно произвести эту операцию и вручную. Мы будем приводить двоичные коды команд: знание лишним не бывает.

Что-то, возможно, будет вам непонятно; не переживайте. После основных сведений о командах пойдет рассказ об их использовании, и в нем по мере надобности будут разъясняться запутанные вещи. Впрочем, не обязательно дожидаться разъяснений: воспользуйтесь отладчиком (DDT или SID; входят в пакеты CP/M и ассемблера) — это прекрасный инструмент, позволяющий вооочию и во всех тонкостях наблюдать процесс исполнения команд.

Начинающим осваивать ассемблер очень мешает отсутствие привычных чисел с плавающей запятой, символьных переменных и т. п. Тут уж ничего не поделаешь — процессор оперирует только с байтами и вариантов интерпретации их содержимого очень мало: целое число без знака (от 0 до 255) или со знаком (от —128 до 127). Впрочем, данные в виде пары байтов тоже иногда допустимы, но содержать они могут только целые числа без знака (от 0 до 65535). Все остальные виды данных приходится моделировать программисту.

Далее мы будем использовать следующие обозначения:

R, R1, R2 — один из регистров A, B, C, D, E, H, L;

RR, RR1, RR2 — одна из регистровых пар BC, DE, HL или SP;

N — 8-разрядное число от 0 до 255;

NN — 16-разрядное число от 0 до 65535;

M — ячейка памяти, адрес которой записан в регистровой паре HL;

( ) — содержимое ячейки памяти или регистра; например, (HL) — число, записанное в HL;

[ ] — ячейка памяти с адресом, обозначенным в квадратных скобках; например, [(BC)] — ячейка памяти, адрес которой равен числу, записанному в BC;

CC (от Cod Condition) — условие, которое должно быть проверено при выполнении команды.

Команд пересылки данных довольно много, они позволяют производить обмен данными между 8- и 16-разрядными регистрами и ячейками памяти.

Ячейки памяти состоят из 8 битов, но благодаря специальным соглашениям в них можно хранить не только 8-разрядные числа. Если, например, нужно записать в ОЗУ двухбайтовое (16-разрядное) число, его заносят в две последовательно расположенные ячейки, причем младший байт числа (разряды 0—7) помещается в ячейку с меньшим адресом (некоторые типы процессоров имеют обратный порядок размещения байтов числа).

Командам пересылки данных все равно, что пересылать, для них важна только длина числа. В зависимости от типа команды перемещаются байт или два байта.

Для пересылки байта пользуются командами однобайтовых пересылок. Они могут занести константу в любой 8-разрядный (байтовый) регистр (непосредственная загрузка); скопировать содержимое любого байтового регистра в любой другой; переписать содержимое аккумулятора в память и загрузить аккумулятор из памяти, причем адрес в памяти может быть указан в команде числом (прямая адресация) или задаваться именем одной из регистровых пар BC, DE или HL (косвенная адресация; в регистровой

N п/п	Мнемоника и длительность выполнения	Код	Действие
1	MVI R,N 7 тактов	0 0 R R R 1 1 0 N N N N N N N N	Загрузка 8-разрядного числа в байтовый регистр
2	MVI M,N 10 тактов	0 0 1 1 0 1 1 0 N N N N N N N N	Загрузка 8-разрядного числа в ячейку памяти, адрес которой записан в HL
3	MOV R1,R2 5 тактов	0 1 R1 R1 R1 R2 R2 R2	Пересылка байта из регистра в регистр
4	MOV M,R 7 тактов	0 1 1 1 0 R R R	Пересылка содержимого байтового регистра в ячейку памяти, адресуемую через HL
5	MOV R,M 7 тактов	0 1 R R R 1 1 0	Пересылка содержимого ячейки памяти, адресуемой через HL, в байтовый регистр
6	STAX B 7 тактов STAX D 7 тактов	0 0 0 0 0 0 1 0 0 0 0 1 0 0 1 0	Пересылка содержимого аккумулятора в ячейку памяти, адресуемую через BC или DE
7	LDAX B 7 тактов LDAX D 7 тактов	0 0 0 0 1 0 1 0 0 0 0 1 1 0 1 0	Пересылка содержимого ячейки памяти, адресуемой через BC или DE, в аккумулятор
8	LDA NN 13 тактов	0 0 1 1 1 0 1 0 N N N N N N N N N N N N N N N N	Загрузка в аккумулятор содержимого ячейки памяти с адресом NN
9	STA NN 13 тактов	0 0 1 1 0 0 1 0 N N N N N N N N N N N N N N N N	Загрузка в ячейку памяти с адресом NN содержимого аккумулятора
10	IN N 10 тактов	1 1 0 1 1 0 1 1 N N N N N N N N	Загрузка содержимого порта N в аккумулятор
11	OUT N 10 тактов	1 1 0 1 0 0 1 1 N N N N N N N N	Загрузка содержимого аккумулятора в порт N

паре должен быть записан нужный адрес). Наконец, возможны непосредственный обмен данными между любым регистром и памятью и непосредственная загрузка константы в память (при этом адрес в памяти должен задаваться содержимым HL). Есть еще две команды ввода-вывода, которые тоже можно отнести к этой группе; одна записывает содержимое аккумулятора в порт (портами называют ячейки из адресного пространства ввода-вывода, чтобы подчеркнуть их отличие от ячеек памяти), другая загружает аккумулятор из порта. Номер порта (от 0 до 255) указывается непосредственно в команде.

Все байтовые команды пересылки данных приведены в таблице.

1. Загрузка 8-разрядного числа в байтовый регистр. Мнемоника команды имеет вид

```
MVI R,N
```


и происходит от слов *move immediate* (непосредственная загрузка) (впрочем, уверенности, что именно эти слова имели в виду специалисты фирмы Intel, разработавшие мнемоники, нет; но мы все же будем приводить «расшифровки» — они очень облегчают запоминание).

Отметим кстати, что мнемоника команд процессора КР580ВМ80А выглядит не совсем удачной, в частности сильно различается для сходных по смыслу команд, но тут ничего не поделаешь. В мнемониках команд более современных процессоров типа 8086, Z80 этого недостатка уже нет, а нам придется напрячь свою память.

При ее выполнении байт N записывается в байтовый регистр.

Рассмотрим машинное представление этой команды, т. е. ее код. В нем должны содержаться идентификация самой команды и используемого регистра, а также загружаемый байт N. Все это умещается в два байта (выпишем их побитно):

0 0 R R R 1 1 0
N N N N N N N N

Первый кодирует команду (биты 7, 6, 2, 1, 0) и регистр (биты 5, 4, 3), во втором записано число N. Регистры здесь и далее кодируются следующим образом:

A — 111
B — 000
C — 001
D — 010
E — 011
H — 100
L — 101

Таким образом, код, например, команды MVI A,1 имеет вид

00111110
00000001

или в 16-ричном виде

3E
01

(далее мы будем, как это принято, прибавлять к записи двоичных чисел B, а 16-ричных — H: 00111110B, 3EH).

Эта команда выполняется за 7 тактов процессора. Если в нашей машине один такт требует 0,4 мкс, то вся команда выполнится за $7\times0{,}4=2{,}8$ мс.

Продолжительность выполнения команды зависит от числа производимых обращений к памяти (их еще называют машинными циклами). Первое обращение длится 4 такта, последующие — по 3. В данном случае для выполнения команды нужно считать только два байта ее кода, что и требует $4+3=7$ тактов.

2. Загрузка 8-разрядного числа в ячейку памяти, адрес которой записан в регистровой паре HL. Мнемоника команды:

```
MVI M,N
```

Поскольку предполагается, что адрес M находится в HL, не пытайтесь найти его в коде команды. Получив в качестве команды последовательность 00110110B, процессор использует число из HL как адрес ячейки памяти и записывает в нее второй байт кода команды.

Если, например, в HL содержалось число 3456H, то при выполнении команды

00110110B=36H
10000000B=80H

(т. е. MVI M,80H) содержимое ячейки с адресом 3456H станет равным 80H.

Отметим, что «ячейка» M во многих командах, не только в MVI, может использоваться аналогично байтовым регистрам. Она даже кодируется очень похоже; в приведенном выше столбце обозначений регистров после регистра L с кодом 101 можно было бы добавить M с кодом 110 (сравните коды команд MVI M,N и MVI R,N).

3. Пересылка байта из регистра в регистр. Мнемоника команды:

```
MOV R1,R2
```

При ее выполнении (R2) (напоминаем, что скобками мы обозначаем содержимое, в данном случае содержимое регистра R2) пересылается в регистр R1. При этом само (R2) не изменяется.

Биты 7 и 6 ее кода задают тип команды, а биты 5, 4, 3 и 2, 1, 0 — названия регистров R1 и R2 (таким же образом, как и в команде MVI R,N). К примеру, код команды MOV A,L равен 01111101B=7DH.

4. Пересылка содержимого байтового регистра в ячейку памяти, адресуемую через HL. Мнемоника этой команды имеет вид

```
MOV M,R
```

Для примера рассмотрим команду MOV M,B (код 01110000B=70H). Если в HL хранится число 1234H, а в регистре B — 71H, то после ее выполнения в ячейке памяти с адресом 1234H будет записано число 71H; содержимое регистров B, H, L не изменится.

5. Пересылка содержимого ячейки памяти, адресуемой при помощи HL, в байтовый регистр. Мнемоника и код этой команды:

```
MOV R,M
```

Для примера рассмотрим команду MOV C,M (код 01001110B=4EH). Если в HL содержится число 5678H, а в ячейке памяти с адресом 5678H записано 55H, то после ее выполнения в регистре C окажется число 55H; (HL) и (5678H) не изменяются.

При однобайтных пересылках ячейки памяти можно адресовать и через регистровые пары BC и DE, но только в том случае, когда данные пересылаются в аккумулятор или из него.

6. Пересылка (A) в ячейку памяти, адресуемую через регистровую пару BC или DE. Мнемоника команд:

```
STAX B
STAX D
```

(от *store accumulator* — сохранить значение аккумулятора; буква X присутствует в мнемониках команд, оперирующих с регистровыми парами).

Пусть (BC)=3210H, (DE)=789AH, (A)=53H. После выполнения команд

```
STAX B
STAX D
```

ячейки памяти с адресами 3210H и 789AH будут содержать число 53H, содержимое регистров A, B, C, D и E не изменится.

7. Пересылка содержимого ячеек памяти, адресуемых через регистровые пары BC и DE, в аккумулятор. Мнемоника команд:

```
LDAX B
LDAX D
```

Пусть (BC)=5432H, (DE)=4321H, (5432H)=51H, (4321H)=72H. Тогда после выполнения команды LDAX B содержимое ячейки памяти с адресом 5432H будет помещено в аккумулятор, т. е. (A) станет равно 51H. Если теперь выполнить команду LDAX D, (A) станет равно 72H; (BC), (DE), (5432H), (4321H) не изменяются.

8. Загрузка из ячейки памяти с адресом NN в аккумулятор. Мнемоника команды:

```
LDA  NN
```

Если, например, (9ABCH)=21H, то после выполнения команды LDA 9ABCH, имеющей код

00111010B=3AH
10111100B=BCH
10011010B=9AH

в регистре A будет записано 21H.

9. Загрузка содержимого аккумулятора в ячейку памяти. Мнемоника команды:

```
STA NN
```

При выполнении, например, команды STA 7FECH, код которой имеет вид

00110010B=32H
11101100B=ECH
01111111B=7FH

(A) будет помещено в ячейку памяти с адресом 7FECH.

10. Загрузка аккумулятора из порта N. Мнемоника команды:

```
IN  N
```

(от *input* — ввод).

Напоминаем, что команды ввода-вывода обращаются не к обычным ячейкам памяти. Они имеют дело скорее с внешними устройствами, наблюдать за которыми с помощью отладчика невозможно и даже опасно.

11. Загрузка порта N в аккумулятор. Мнемоника команды:

```
OUT N
```

(от *output* — вывод).

Продолжение следует.

А. ПИУНОВ
г. Каменск-Уральский

Графические редакторы для ПЭВМ «Нейва» («Корвет»)

Думаю, все работавшие на компьютере сталкивались с графическими редакторами и знают, что они предназначены для создания и редактирования изображений. Чаще всего ими пользуются, когда нужно вставить в программу иллюстрации. Я серьезно занялся графическими редакторами, когда понадобилось нарисовать картинки для только что законченной игры.

Когда на «Нейве» еще не было сетевой операционной системы «Корнет», я сделал себе графический редактор на Бейсике, чтобы им можно было пользоваться на РМУ и вставлять созданные им изображения в Бейсик-программы. Но вскоре изучил язык Си и стал осваивать GRAF, рисунки из которого можно было сразу использовать в Си-программах (тогда в графической библиотеке Си еще не было чтения иллюстраций, нарисованных в ABRISe). Затем освоил ABRIS и GRIN. Пытался использовать и GROT, но его возможности уж очень скромны.

Как видите, графических редакторов для «Нейвы» существует немало. А какой самый лучший — это зависит от ваших нужд. Попытаюсь охарактеризовать их хотя бы кратко, чтобы помочь сделать выбор.

Рисование пером, создание линий, прямоугольников, окружностей работают во всех редакторах примерно одинаково, но у ABRISa есть существенное преимущество: восемь размеров и различные формы пера. Возможности закраски в ABRISe также шире: можно «перекрасить» цвет, залить до любого контура. К тому же закраску и заливку можно производить узором. В GRAFe можно закрашивать выбранным цветом (до границы того же цвета), в GRINe — перекрасить участок одного цвета в другой, в GRISe — закрасить любым цветом до заданного контура.

Лупа в ABRISe позволяет увидеть изображение в масштабе 8:1 и 4:1, в GRINe — 4:1; зато в GRINe увеличенное изображение появляется моментально, а в ABRISe сначала записывается на диск основная картинка, а уже потом появляется увеличенное изображение.

Вставить в рисунок текст можно в трех редакторах. В ABRISe буквы могут быть восьми размеров, любого цвета (или узора), в восьми ориентациях и к тому же можно самому создать шрифт или отредактировать уже имеющийся. Шрифт в GRINe (здесь его можно вставлять в изображение в четырех ориентациях) и в GRAFe схож со шрифтом знакогенератора.

Копировать изображения в ABRISe весьма хлопотно (особенно если надо много копий): нужно записать копируемую область на диск, а затем воспроизвести в нужном месте. GRIN и GRAF позволяют копировать изображение сразу же. GRIN предоставляет возможности для масштабирования: можно фрагмент любого размера увеличить или уменьшить, а также перевернуть или сделать зеркальное отображение. GRAF может растянуть изображение в два раза вправо или вниз и сжать в два раза сверху или слева.

Число доступных цветов обозначено в таблице. Знающих «Нейву» эта цифра в столбце ABRISa удивит: как из восьми возможных цветов сделать 30? Довольно просто: расположив точки различных цветов в шахматном порядке, получили производные цвета (конечно, на монохромном дисплее этот прием не дает эффекта). К тому же в ABRISe можно не только пользоваться уже имеющимися узорами, но и создавать собственные (сохраняя их затем на диске).

«Мышью» можно пользоваться в ABRISe и GROTe. Возможно, скоро появится новая версия GRINa, допускающая это. К сожалению, отечественные «мышки» имеют такое качество, что предпочтение следует отдать клавиатуре. И все же некоторые функции выполнять «мышью» намного быстрее.

Вывести изображение на принтер можно из всех редакторов, кроме GRISa, из ABRISa — даже на цветной матричный принтер. Если принтер простой, то изображение получается черно-белое, но, поскольку печать производится в четыре прохода, точки получаются различной яркости. GRIN может печатать как позитивное, так и негативное изображение, имитируя 4 или 8 цветов. Точка изображения на принтере состоит из нескольких точек, расположенных так, чтобы передать яркость точки на экране, что позволяет печатать в один проход. В GRAFe можно выбрать режим печати: негатив, позитив, 1, 3 и 6 проходов. Точка экрана соответствует точке принтера. Недостатком является очень малая скорость печати. GROT соответственно своим двум цветам печатает изображения в один проход.

Все редакторы позволяют хранить изображение на диске. ABRIS может записать на диск всю картинку или только часть, при этом сжимая их до минимальных размеров. GRIN может сохранить картинку полностью: один к одному 3 слоя по 16 Кбайт, со сжатием или окно. GRAF записывает и весь экран, и окно всегда со сжатием. GRIS хранит изображение в виде координат его элементов.

Несколько слов об использовании рисунков в других графических редакторах и в программах. Так как GRAF написан на языке Си, то нарисованные в нем изображения сразу можно передать в Си-программу. Картинку из ABRISa можно оставить на экране и, запустив GRIN, отредактировать или сохранить ее на диске. Чтобы использовать рисунки из GRINa в GRAFe (а также в Си-программах) и наоборот, мне пришлось написать программы GRINSI и SIGRIN; сейчас они поставляются вместе с редактором GRIN.

Если из текста программы редактора GRIS выделить подпрограмму просмотра рисунков и вставить в Бейсик-программу, то в ней можно будет использовать созданные в GRISe изображения.

Попробую дать интегральную оценку. Безусловно, самый мощный из редакторов — ABRIS. У него большие возможности для рисования изображений, я бы даже сказал, художественных картинок (об этом говорит уже фирменная заставка). Широкий выбор цветов и узоров (который к тому же можно расширять самому) позволяет получить качественное цветное изображение на экране, что очень ценится в игровых программах. В числе достоинств и единообразие управления всеми функциями, так что научиться пользоваться ими несложно; впрочем, создание шрифтов или узоров вызывает поначалу затруднения. Недостатки — отсутствие функций копирования и масштабирования фрагментов изображения, а также частое обращение к диску (функции редактора находятся в файле оверлея и при каждом обращении к ним подгружаются).

Второе место занимает GRIN. В нем можно быстро нарисовать изображение, подогнать под нужный размер и напечатать или сохранить на диске. Скорость исполнения команд высока, освоение редактора происходит легко, работать удобно. Рисунки из него можно использовать в программе на любом языке, если написать подпрограмму, читающую картинку с диска и записывающую в ГЗУ.

GRAF написан на Си, поэтому его возможности определяются возможностями функций из графической библиотеки Си. Редактор имеет встроенный краткий Help (подсказку), но не имеет меню, поэтому нужно либо запомнить его команды, либо постоянно пользоваться подсказкой. В целом редактор хороший, хотя нарисовать в нем картинку довольно сложно.

GRIS написан на Бейсике, поэтому его возможности очень скромны. В нем можно создавать лишь несложные изображения. GRIS можно порекомендовать только работающим на РМУ без использования СОС «Корнет».

GROT был создан для иллюстрирования текстов графиками, чертежами и простыми рисунками. Для его освоения потребуется всего несколько минут, но это следствие весьма ограниченных возможностей. Об использовании его изображений в программах ничего сказать не могу, так как даже не пытался выяснить это, имея более мощные редакторы.

Характеристики графических редакторов

	ABRIS	GRIN	GRAF	GRIS	GROT
Перо	++	+	+	+	+
Линия	++	+	+	+	+
Прямоугольник	++	+	+	+	+
Окружность	++	+	+	+	+
Закраска	++	+	+	+	+
Лупа	+	+	–	–	–
Текст	++	+	+	–	–
Копирование	+–	++	+	–	–
Масштабирование	–	++	+	–	–
Цвета/узоры	30/18	8/8	8/8	8/8	2/8
Печать	+	++	+	–	+
"Мышь"	+	–	–	–	+
Размер картинок на диске, Кбайт	<48	48/<48	<48	<=8	16
Размер редактора на диске, Кбайт	110	12	40	12	16
Язык, на котором написан редактор	Си	Ассемблер	Си	Бейсик	Ассемблер

Условные обозначения: (–) – функции нет; (+) – функция есть; (++) – функция реализована очень удобно.

Все графические редакторы поставляются производственным объединением «Октябрь» (623400, ГСП-79); акционерным обществом «Хорда» (623400, Екатеринбургская обл., г. Каменск-Уральский, а/я 1); малым предприятием «Микс» (119899, Москва, Ленинские горы, МГУ, НИИЯФ, «Микс»).

А. ДМИТРИЕВ

Что может принтер «Корвета»

Печатающие устройства, имеющиеся в каждом компьютерном классе, зачастую используются лишь для получения листингов программ. Между тем современный принтер — универсальный инструмент, который может хорошо поработать в интересах не только уроков информатики, но и всей школы.

Рассмотрим возможности принтеров на примере индийского CPF-H80, которым преимущественно комплектуются «Корветы» (многие другие модели управляются так же). В распоряжении его владельца имеются большой выбор шрифтов, подстрочные и надстрочные символы, возможности управления режимами печати и использования самостоятельно созданных символов, отсутствующих в стандартном наборе символов ASCII.

Начальную установку режима работы принтера можно выполнить с помощью кнопок принтера и набора переключателей на его передней панели. Но нас больше интересует возможность управления режимами печати программным путем с помощью так называемых ESC-последовательностей (напомним, что буквами ESC обозначается символ с кодом 27).

В качестве примера использования принтера приведем фрагмент текста и Бейсик-программу печати этого текста.

```
2.4. АЛГЕБРАИЧЕСКИЕ
     УРАВНЕНИЯ

2.4.2. Общие теоремы.

Основная теорема алгебры:
Каждое алгебраическое уравнение n-й степени
 X^n+a1X^n-1+...+an-1X+an=0,
коэффициенты которого ai (i=1,2...n)-действительные
или комплексные числа, имеет ровно n корней, если
k-кратный корень считать за k корней.
```

```
10 DEFSNG A,S
20 REM ********** ОПРЕДЕЛЕНИЕ УПРАВЛЯЮЩИХ КОДОВ **
30 SN=CHR$(27)+"@"  :REM Начальная инициализация ПУ *
40 SU=CHR$(27)+"!"  :REM Универсальный переключатель*
50 SR=SU+CHR$(32)   :REM Печать расширенных символов*
60 SB=SU+CHR$(8)    :REM "Жирный шрифт"             *
70 SI=SU+CHR$(64)   :REM Курсив (режим ITALIC)      *
80 SP=SU+CHR$(128)  :REM Печать с подчеркиванием    *
90 RU=SU+CHR$(0)    :REM Восстановление нормального *
100 REM                        режима               *
110 SH=CHR$(27)+"S"+CHR$(0) :REM Надстрочн.символы  *
120 SL=CHR$(27)+"S"+CHR$(1) :REM Подстрочн.символы  *
130 RT=CHR$(27)+"T"         :REM Выключение         *
140 REM        надстрочных и подстрочных символов   *
150 REM ************************************************
160 LLIST SN;CHR$(27);"l";CHR$(10):REM Начальная
170 REM инициализация принтера и смещение текста
180 REM                  вправо на 10 позиций
190 LLIST SR;"   2.4. АЛГЕБРАИЧЕСКИЕ"
195 LLIST "           УРАВНЕНИЯ"
200 LLIST
210 LLIST SB;"2.4.2. Общие теоремы."
220 LLIST:LLIST SI;"Основная теорема алгебры:"
230 LLIST RU;"Каждое алгебраическое уравнение n";
                                SH;"-й";RT;" степени"
240 LLIST " X";SH;"n";RT;"+a";SL;"1";RT;"X";SH;
                                "n-1";RT;"+...+a";
250 LLIST SL;"n-1";RT;"X+a";SL;"n";RT;"=0,"
260 LLIST "коэффициенты которого a";SL;"i";RT;
270 LLIST " (i=1,2...n)-действительные"
280 LLIST "или комплексные числа, имеет ";SP;"ровно
                                n корней";
290 LLIST RU;", если"
300 LLIST "k-кратный корень считать за k корней."
310 LLIST
```

Команда переключения шрифтов ESC ! *n* в зависимости от значения *n* устанавливает следующие шрифты:

```
n=0   (&B00000000) нормальная печать (шрифт "PICA")
n=1   (&B00000001) шрифт "ELITE"
n=4   (&B00000100) сжатый шрифт
n=8   (&B00001000) "жирный" шрифт
n=16  (&B00010000) режим печати "двойной удар"
n=32  (&B00100000) расширенные символы
n=64  (&B01000000) курсив (шрифт "ITALIC")
n=128 (&B10000000) печать с подчеркиванием.
```

В ряде случаев возможно использование комбинации режимов. Так, ESC-последовательность ESC ! 224 (224=&B11100000) задает печать широких наклонных символов с подчеркиванием. Из Бейсика этот режим можно включить командой

```
LPRINT CHR$(27);"!";CHR$(224)
```

Перечислим еще ряд важных команд управления печатью.

```
ESC A n  (CHR$(27);"A";CHR$(n))
```

задает интервал между строками (в точках, или пунктах; один пункт равен 1/72 дюйма);

```
ESC l n  (CHR$(27);"l";CHR$(n))
```

задает номер позиции левой границы печати;

```
ESC Q n  (CHR$(27);"Q";CHR$(n))
```

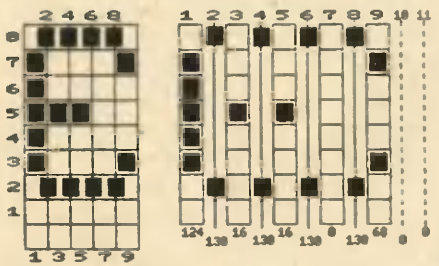
задает номер позиции правой границы печати;

```
SB (CHR$(8))
```

— возврат на один символ (позволяет печатать два символа в одной позиции).

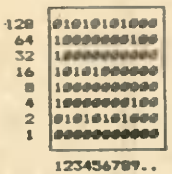
В связи с возрастанием роли национальных языков обострилась проблема отсутствия на пишущих машинках национальных литер. Возможность кодировки символов на матричных принтерах позволяет обойти эту проблему.

Каждый символ, выводимый на печать, задается матрицей 8×11. Например, украинская буква ■ * формируется матрицей, представленной на рисунке.



Обратите внимание, что сетка литеры на рисунке имеет 9 строк, а матрица — только 8. Дело в том, что для задания символа могут использоваться только 8 верхних или 8 нижних (для букв с «хвостиком») строк сетки. Выбор верхних или нижних строк осуществляется специальным кодом. Следует также иметь в виду, что в каждой строке матрицы единицы должны обязательно чередоваться с нулями, а два правых столбца резервируются для обеспечения интервала между символами.

Полученная матрица



вводится в принтер по столбцам. Так, код столбца 1 матрицы равен 124 (&B01111100), столбца 2 — 130 (&B100000010), столбца 3 — 16 (&B00010000) и т. д.

Строка с приказом о вводе в принтер нового рисунка символа выглядит так:

```
LPRINT CHR$(27);"&";CHR$(0);CHR$(69);CHR$(69);
       CHR$(139);CHR$(124);CHR$(130);CHR$(16); ...
```

Здесь первые три символа — признак начала команды, определяющей новый символ (ESC & 0); следующие два — начальный и конечный коды перепрограммируемых символов (если задается изображение для одного символа, то начальный и конечный коды совпадают); CHR$(139) — тот самый код выбора верхних 8 строк матрицы, о котором мы уже говорили (нижние строки задает код CHR$(11)). Остальные элементы строки — коды столбцов рассмотренной выше матрицы.

Заметим, что новый символ можно создать только вместо уже имеющихся символов в диапазоне кодов 33—126, т. е. вместо цифр, знаков и букв латинского алфавита.

```
10 LPRINT CHR$(27);"@";:REM Начальная инициализация
                                принтера
20 REM Подготовка принтера к загрузке нового шрифта
30 LPRINT CHR$(27);"%";CHR$(1);CHR$(0);:REM        К
40 LPRINT CHR$(27);":";CHR$(0);CHR$(0);CHR$(0);:REM
50 REM ХХХХХХХХХХХХХХХХХХХХХХХХХХХХХХХХХХХХХХХХХХХ
60 REM Загрузка новых символов-------------------
70 LPRINT CHR$(27);"&";CHR$(0);CHR$(69);CHR$(69);
   CHR$(139);CHR$(124);CHR$(130);CHR$(16);CHR$(130);
     CHR$(16);CHR$(130);CHR$(0);CHR$(130);CHR$(68);
     CHR$(0);CHR$(0);  :REM Перекодировка символа "E"
80 LPRINT CHR$(27);"&";CHR$(0);CHR$(101);CHR$(101);
     CHR$(139);CHR$(28);CHR$(34);CHR$(8);CHR$(34);
       CHR$(8);CHR$(34);CHR$(0);CHR$(34);CHR$(20);
     CHR$(0);CHR$(0);  :REM Перекодировка символа "e"
90 LPRINT CHR$(27);"&";CHR$(0);CHR$(106);CHR$(106);
       CHR$(139);CHR$(0);CHR$(0);CHR$(162);CHR$(0);
         CHR$(62);CHR$(0);CHR$(162);CHR$(0);CHR$(0);
     CHR$(0);CHR$(0);  :REM Перекодировка символа "i"
100 REM-------------------------------------------
```

В приведенной программе три новых литеры загружаются вместо латинских букв E (код 69), *e* (код 101) и *j* (код 106) соответственно. Разумеется, при наборе текста на мониторе будут отображаться именно «законные» латинские буквы, а новые символы будут печататься вместо них на принтере.

Программа перекодировки символов должна выполняться перед выводом текста на печать. Новые символы сохраняются в памяти принтера до его выключения. Начальную настройку принтера для печати кириллицей выполняют с помощью стандартной утилиты SETPRI (не следует в этом случае использовать программу EPSON).

Комментарий специалиста

Появление таких публикаций можно только приветствовать. Ясное изложение принципов управления принтером CPF-H80 будет полезно многим читателям, желающим пойти по стопам автора статьи и разработать программы печати.

Действительно, возможности принтеров используются далеко не полностью. На них

* К сожалению, в типографии, где набирается «ИНФО», такого символа нет.— *Примеч. ред.*

можно не только печатать школьные документы и материалы для учебных занятий, но и, к примеру, издавать вполне профессионально оформленную школьную газету. Однако так ли необходимо для этого изучать документацию к принтеру, писать, затрачивая недели и месяцы, собственные сервисные программы? Нет. Программы FX, TYPEDOC и PRINT, подготовленные в МП «Микс», позволяют использовать принтер на 100 %. Они имеют удобный интерфейс, работают под управлением ОС CP/M-80.

FX предназначена для распечатки текстовых файлов на принтерах RAVI-8010M, FX-80, FX-850, ComputeMate. Она позволяет задать ширину левого поля, расстояние между строками, тип бумаги (рулон, листовая) и длину листа, нумерацию страниц и т. п., а также любой из допустимых для каждого принтера шрифтов. Выбор файла для печати из каталога диска производится курсором. Если вам нужно изменить в отпечатанном документе одну страницу, вы можете распечатать только ее; можно также распечатывать со сквозной нумерацией документ, записанный в нескольких файлах. Можно, наконец, запомнить заданные параметры, и при следующем вызове FX принтер будет настроен так же, как в прошлый раз.

TYPEDOC позволяет печатать в графическом режиме. Об этой возможности матричных принтеров в статье не упомянуто, а ведь если принтер может печатать изображения, то его можно заставить выводить и буквы больших размеров или необычных начертаний! TYPEDOC (она работает с теми же принтерами, что и FX) предлагает на выбор несколько шрифтов: стандартный (имитация шрифта пишущей машинки), выделенный, курсив, средний. К сожалению, менять шрифты внутри текста нельзя.

За качество, как всегда, приходится платить — TYPEDOC печатает вчетверо медленнее, чем FX.

Если вы хотите выделить в тексте отдельные слова, вам нужно вставить в него управляющие ESC-последовательности (как это сделать, можно узнать из инструкции к текстовому редактору) и распечатать с помощью FX.

PRINT — программа универсальная. К возможностям TYPEDOC и FX она добавляет следующие:

использование принтеров из более широкого набора (в частности, «Роботрон К6311М», СМ 6337);

новые шрифты в графическом режиме (набор математических символов, «рукописный» шрифт и др.);

даже в графическом режиме — переход на другой шрифт внутри текста (это позволяет очень красиво оформлять распечатки, вставлять сложные математические формулы и т. п.);

включение в текст изображений, нарисованных в графическом редакторе ABRIS.

В PRINT имеется также редактор шрифтов, позволяющий создавать собственные наборы символов (в частности, использовать национальные алфавиты).

После того как создан новый шрифт, встает проблема текстового редактора. Не очень-то удобно работать по принципу «j пишем, знак интеграла в уме». Снимает проблему редактор ChiWorker, экранные и принтерные шрифты которого (одновременно может использоваться 16 их наборов) совпадают. Плата за это — большой размер командного файла, из-за чего с ним неудобно работать на РМУ. Для работы в локальной сети с загружаемыми экранными шрифтами нужен компактный «графотекстовый» редактор. Стоимость его создания, по нашим оценкам, около 50 000 рублей. МП «Микс» готово взяться за эту работу, если на редактор будет сделано не менее 100 заявок.

Последняя проблема при использовании национальных алфавитов — обозначения на клавишах. Ее тоже можно решить — при наличии более 100 заказов будет организован выпуск клавиатуры с любой символикой.

А. НЕЧАЕВ

Комментарий от редакции. Как видите, о пользователях неплохо позаботились — существует выбор программ с разнообразными возможностями печати. Так что же, изучать принтер не нужно? Конечно, готовым пользоваться легче, но кто-то ведь должен делать «готовое»! К тому же у каждого пользователя могут возникнуть свои специфические проблемы (к примеру, в редакции пришлось написать программу, позволяющую печатать для нашей типографии качественные, «черные», тексты через изношенную, «серую», ленту). Так что изучать или нет — решайте сами, памятуя, что знания лишними не бывают.

А. РУССКИХ

Что может ячейка FAFB

В Бейсике «Корвета» есть оператор LUT. Выполнить все, что делает он, к тому же экономя память и с большим быстродействием, можно, используя ячейку FAFB (на ПК 8020 аналогичная ячейка имеет адрес FFFB)*.

Выводимый на экран цвет называется физическим цветом, а задаваемый в графических операторах — логическим. Содержимое ячейки FAFB задает соответствие между ними. Обозначим содержимое ее битов C0 (0-й бит), C1 (1-й бит) и т. д. до C7. Число, составленное из C0, C1, C2, C3, задает номер логического цвета; C6, C5, C4 задают физический цвет (C6 — наличие красного, C5 — зеленого, C4 — синего компонентов); C7 регулирует яркость физического цвета (при C7=1 яркость максимальна, при C7=0 — вдвое меньше). Яркость можно рассматривать в качестве компонента физического цвета, и оказывается, что 16-ричное представление содержимого ячейки FAFB первой цифрой кодирует физический цвет, а второй — логический. Очень просто и удобно! Достаточно изменить значение этой ячейки, и сразу изменится вид экрана. В качестве примера — рисование окружности с центром в точке (100,100) физическим цветом 7, логическим цветом 1 и «мгновенный» вывод текста.

```
10 CLS: PCLS
20 POKE &HFAFB,&H71 'на ПК 8020 POKE
                              &HFFFB,&H71
21 'В &H71 7 - физический цвет;
22 '         1 - цвет, указанный
23 '           в графическом операторе.
30 POKE &HFAFB,&H8 'на ПК 8020 POKE
                              &HFFFB,&H8
31 'Здесь цвет символов перекодируется
32 '                        в цвет фона
33 'Логический цвет символов - 8
40 PRINT "-=:* Star system *:=-"
50 CIRCLE (100,100),50,1,,,1.5
60 POKE &HFAFB,&HF8 'на ПК 8020 POKE
                              &HFFFB,&HF8
70 END
```

Теперь — «мгновенный» вывод на экран рисунка. Для этого «погасим» все цвета (обнулим физические цвета), а затем, создав рисунок, восстановим их.

```
10 CLS: PCLS
20 FOR I=&H1 TO 8
30 POKE &HFAFB,I 'на ПК 8020 POKE
                              &HFFFB,I
40 NEXT
50 'Создание рисунка графическими
                              командами
...
200 'Конец рисования
210 FOR I=&H11 TO &H88 STEP &H11
220 POKE &HFAFB,I 'на ПК 8020 POKE
                              &HFFFB,I
230 NEXT: END
```

Сравним программы с использованием LUT и непосредственно ячейки FAFB.

```
10 DEFINT L
20 DIM L(31)
30 FOR I=0 TO 7:
     L(I)=I: NEXT
40 FOR I=8 TO 15:
    L(I)=15: NEXT
50 FOR I=0 TO 8
60 L(I)=0: NEXT
70 LUT L(0)
80 END
```

```
10 'Нет операций
20 'Нет операций
30 'Нет операций
40 'Нет операций
50 FOR I=0 TO 8
60 POKE &HFAFB,I:
                    NEXT
70 'Нет операций
80 END
```

Эти программы маскируют все цвета в цвет фона. Из примера ясно, что использование LUT требует большего времени, хотя, конечно, иногда и LUT может быть удобен.

От редакции. В момент изменения содержимого ячейки FAFB на экране может возникнуть мерцание. Использование LUT позволяет избежать этого.

---

* Этот адрес может меняться в зависимости от версий Бейсика и МикроДОСа.

Учебно-методический центр «Информвидео»

Московский институт повышения квалификации работников образования

приглашает учителей информатики

на курсы по научно-методическим основам преподавания информатики и обучению работе в дисплейном классе

Курсы в объеме 72 ч проводятся в каникулярное время в течение 10 дней: март, июнь, ноябрь 1992 г., январь и т. д. 1993 г.

Обучение проводится на различных видах ВТ: УКНЦ, КУВТ-86, БК-0011, БК-0011М, «Агат 7, 9», «Корвет», «Ямаха», IBM PC/XT/AT/PS-2.

Слушателям курсов передается программно-методический комплекс, включающий сетевые, сервисные и прикладные программы по учебникам А. Г. Кушниренко и В. А. Каймина и методическую литературу.

Приглашает потенциальных пользователей компьютерной телекоммуникационной сети «SCHOOLNET» на курсы абонентов телекоммуникационной сети.

Курсы в объеме 72 ч проводятся 1—10 июля, 21—30 августа, 16—25 декабря 1992 г.:

операционные системы MS-DOS и OS/2, русифицированное телекоммуникационное программное обеспечение;

организация и функционирование сети, работа абонента сети.

Слушателям курсов предоставляется первоочередное право регистрации сетевого адреса и приобретения на льготных условиях аппаратных (модемов) и программных средств абонента сети.

Приглашает управленцев народного образования (директоров, завучей, работников УНО) на курсы «Компьютеры в управлении образованием»

Курсы в объеме 72 ч проводятся в апреле, июле, октябре 1992 г.:

компьютерная грамотность (умения и навыки работы на компьютере);

компьютеризация управления учебным процессом (информационно-справочные системы, программа составления расписания и т. д.).

Предлагает приобрести

Методические пособия:

1. Информатика (теория, методика, задачи).
2. Применение компьютеров IBM PC/XT/AT/PS-2.
3. Работа в дисплейном классе УКНЦ.

Программное обеспечение на все виды ВТ

Слушателям курсов выдается сертификационное удостоверение, необходимое в процессе аттестации.

Слушатели обеспечиваются общежитием гостиничного типа.

Заявки (гарантийные письма) просим высылать по адресу:

125167, Москва, Авиационный пер., д. 6, ком. 205.

Телефон для справок: 151-59-31.

А. ГУСЕВ

Операционная система ФОДОС

1. Архитектура УКНЦ

1.1. Состав класса

Класс УКНЦ, как правило, состоит из 13 машин: 12 рабочих мест учеников (в дальнейшем — РМУ) и одного рабочего места преподавателя (в дальнейшем — РМП). Все машины объединены в кольцевую локальную сеть (ЛВС). ЭВМ в составе локальной сети называется станцией. Максимальное число машин, которое можно объединить в сеть,— 64. Однако большинство программного обеспечения поддерживает не более 15—16 станций.

РМУ состоит из:

ЭВМ «Электроника МС0511» (УКНЦ);

видеомонитора (как правило, монохромного «Электроника МС6105»).

РМП состоит из:

ЭВМ «Электроника МС0511» (УКНЦ);

видеомонитора (как правило, монохромного «Электроника МС6105»);

контроллера накопителя на гибких магнитных дисках, двойной плотности формата записи MFM (КНГМД);

накопителя на гибких магнитных дисках НГМД-6022 (40 дорожек 400 Кбайт, 800 блоков) или НГМД-5309 (82 дорожки 820К байт, 1640 блоков);

печатающего устройства (ROBOTRON, D100, D100M, СМ-6337 и др.).

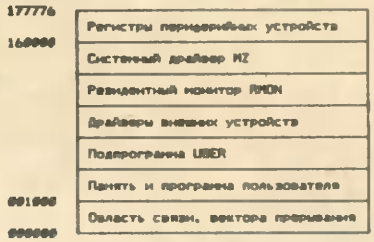
1.2. Архитектура УКНЦ

В основу архитектуры УКНЦ положен модульно-магистральный принцип. Обмен информации производится по 16-разрядной системной магистрали типа Q-bus. ЭВМ УКНЦ является двухпроцессорной машиной. В ее состав входят центральный и периферийный процессоры.

Тип центрального (ЦП) и периферийного (ПП) процессоров — К1801ВМ2. Разрядность процессоров — 16 бит. ОЗУ ЦП составляет 64 Кбайта, ОЗУ ПП — 32 Кбайта, ВидеоОЗУ состоит из трех планов по 32 Кбайта.

1.3. Распределение памяти ЭВМ при работе с операционной системой

Операционная система работает только с ОЗУ ЦП. Доступ к другим частям ОЗУ (ПП и видеоОЗУ) осуществляется с помощью драйверов. Далее приводится схема распределения памяти ЦП.



2. Назначение операционной системы

Операционная система (ОС) является базовой для ЭВМ. С помощью ОС пользователь общается с ЭВМ, загружает и запускает на выполнение прикладные программы. В функции ОС входит и обмен данными с драйверами внешних устройств.

Монитор — это основная программа ОС (не путайте с видеомонитором или дисплеем, которые являются физическими устройствами отображения информации).

Наиболее часто встречающиеся на УКНЦ мониторы: *RT11SJ, RT11CD, FMONSJ*. Две последние буквы в имени определяют тип монитора. *SJ*-монитор — однозадачный монитор реального времени. *CD*-монитор — модификация монитора *SJ*. *CD*-монитор обеспечен дополнительными сервисными сообщениями, что значительно облегчает работу на ЭВМ. Внешне для пользователя они отличаются приглашением к работе. Монитор *SJ* в качестве приглашения к работе выдает ТОЧКУ, а монитор *CD** — имя устройства, выбранного по умолчанию.

Пример:

● *RUN BASIC* — командная строка для *SJ*-монитора.

MZ⟩ *RUN BASIC* — командная строка для *CD*-монитора.

Мониторы *XM* и *TS* на УКНЦ не применяются.

2.1. Файловая система

Все программы и данные хранятся на магнитных дисках или магнитофонных лентах в виде файлов (далее рассматривается наиболее распространенное внешнее запоминающее устройство — накопитель на гибких магнитных дисках, НГМД). Файл имеет имя, состоящее из больших латинских букв или цифр, причем длина имени не должна превышать 6 символов. Дополнительная информация о файле содержится в расширении имени файла, состоящего также из больших латинских букв или цифр. Расширение файла, длиной не более 3 символов, отделяется от имени точкой. Например:

RT11CD.SYS — файл, в котором записан монитор. Его имя *RT11CD*, а расширение *SYS*.

COD.SAV — программа вывода на экран кода клавиш.

Информация о размещении файлов на диске хранится в строго определенном месте, называемом каталогом. При запуске программы на выполнение или выводе содержимого файла, например, на принтер ОС сначала анализирует информацию о файле, хранящуюся в каталоге, а затем осуществляет считывание. Информация в каталоге содержит имя файла (с расширением), его размер, местоположение и еще некоторые реквизиты.

2. Загрузка операционной системы

Включите ЭВМ, видеомонитор и дисковод. Вставьте системный диск в верхний карман дисковода. На экране появится меню:

```
ЗАГРУЗКА
--------
1 - диск              (0-3): 0
2 - касета ПЗУ        (1,2): 1
3 - сеть
4 - стык С2
5 - магнитофон
6 - отладка
7 - тестирование
```

В левом углу, над цифрой 1, мигает белый квадрат — курсор, указывающий то место, куда будет выводиться символ. Он перемещается по экрану клавишами управления курсором. Нажмите один раз на клавишу ⟨↓⟩. Курсор переместится на строку вниз. Нажмите на клавишу ⟨ВВОД⟩. Загорится красный индикатор выбора кармана дисковода, свидетельствующий о начале чтения ОС с диска. На экран будут выводиться сообщения о ходе загрузки ОС.

В конце загрузки запускается программа *DAY*, сообщающая о последнем дне работы и запрашивающая текущую дату. Нажимая на клавишу ⟨ПРОБЕЛ⟩, установите текущее число, день недели, месяц и год. Нажатие на ⟨ПРОБЕЛ⟩ увеличивает текущую дату на одно число. Если вы лишний раз нажали на ⟨ПРОБЕЛ⟩, то нажатием на клавишу ⟨—⟩ можно вернуть предыдущую дату. Клавишами ⟨*L*⟩ и ⟨*H*⟩ дата меняется по месяцам.

* Далее в тексте мы будем обращаться к монитору *RT 11CD*.

В процессе работы программа пользователя обменивается информацией с внешними устройствами, читает и записывает данные, например, на магнитный диск, не обращаясь при этом к физическим адресам внешних устройств, так как это обращение исключает возможность переноса ее на другой тип ЭВМ.

Например, у УКНЦ и ДВК, имеющих один и тот же процессор, подключение внешних устройств, например дисководов, разное. Однако практически все программы, предназначенные для ДВК (база данных *RDB*, электронные таблицы, текстовые редакторы да и сама операционная система), работают и на УКНЦ. Посредником между программой и внешними устройствами выступают драйверы внешних устройств, обращающиеся непосредственно к физическим адресам и представляющие собой небольшую и, как правило, несложную программу. При переносе операционной системы с одного типа машин на другой требуется изменить только драйверы, а самая большая и сложная программа — монитор операционной системы — остается прежней (для ЭВМ серии ДВК, исключением является ЭВМ БК-0011, так как ее нестандартное распределение памяти не позволяет использовать стандартный монитор операционной системы). Единственное устройство, к которому обращается операционная система напрямую,— это терминал. Под одним именем «терминал» понимаются два устройства — дисплей для вывода информации и клавиатура для ввода. На всех машинах типа ДВК дисплей и клавиатура подключены через одни и те же регистры (за исключением БК-0011).

При подключении к машине принтеров *D*100M, СМ-6337 или *Epson* целесообразно установить автозапуск программы *D*100M при загрузке операционной системы. Порядок установки автозапуска см. в п. 4.5. После установки нужной даты нажмите на клавишу ⟨ВВОД⟩. После этого (в случае *CD*-монитора) на экране появится приглашение ОС:

```
MZ0>
```

информирующее о том, что устройством по умолчанию является верхний (нулевой) карман дисковода.

В случае неисправности верхнего кармана дисковода можно загрузить ОС с нижнего. Для этого после включения машины надо:

а) переместить курсор в строку: 1 — диск (0—3) : 0;

б) нажать на клавишу ⟨→⟩;

в) нажать на клавишу ⟨1⟩ (т. е. указать номер кармана, с которого будет производиться загрузка ОС);

г) нажать на клавишу ⟨ВВОД⟩.

После этого начнется загрузка ОС с нижнего кармана дисковода.

Приглашение ОС к работе будет теперь иметь вид:

```
MZ1>
```

Устройством по умолчанию теперь является нижний (первый) карман дисковода.

3. Команды монитора операционной системы

Пользователь общается с операционной системой посредством ввода команд, которые обрабатываются специальной программой обработки команд *KMON* (*Keyboard Monitor*) — клавиатурный монитор. *KMON* принимает команду пользователя, интерпретирует ее и выполняет соответствующие действия. Для ввода команды необходимо набрать ее на клавиатуре (при этом набираемые символы высвечиваются на экране дисплея) и нажать на клавишу ⟨ВВОД⟩.

Команды можно набирать как в полном варианте, так и в сокращенном. Например, для просмотра каталога диска нужно ввести команду ***DIRECTORY*** (полная команда) и нажать на клавишу ⟨ВВОД⟩, а можно набрать только первые три символа ***DIR*** и нажать на клавишу ⟨ВВОД⟩. Далее будут приводиться полные варианты команд, а в скобках — их допустимые сокращения, например: ***DIRECTORY (DIR)***.

В командах могут применяться ключи. Они ставятся после команды или имени файла и отделяются символом /, например: *DIR/VOL* — просмотр каталога диска с выводом информации о пользователе. Ключи можно совмещать, т. е. в одной команде может быть использовано несколько ключей, например: ***DIR/FU/VOL/BLO MZO***:

3.1. Просмотр каталога диска

DIRECTORY (DIR) — команда просмотра каталога диска. Для просмотра содержимого диска следует набрать ***DIR*** и нажать на клавишу ⟨ВВОД⟩:

```
MZ0>DIR
```

Внимание! Приглашение ***MZ0***⟩ *выдает операционная система и пользователем не набирается.*

Если операционная система была загружена с первого кармана, то приглашение будет иметь вид: ***MZ1***⟩.

После введения этой команды произойдет обращение к верхнему карману дисковода (загорится световой индикатор выбора нулевого кармана дисковода) и на экран дисплея будет выведен каталог диска. Например:

```
25-Dec-91
SWAP    .SYS     26  02-Sep-86    RT11CD.SYS     70  23-Aug-86
LP      .SYS      2  01-Jun-89    TT     .SYS     2  01-Jan-86
MC      .SYS      6  10-Aug-87    MZ     .SYS     4  15-Jan-90
DIR     .SAV     19  10-Aug-87    PIP    .SAV    29  15-Jan-90
DUP     .SAV     45  10-Aug-87    DAY    .SAV     4  31-Mar-87
STARTB.COM        1  01-Sep-90    BASIC  .COM     1  01-Feb-88
BASIC  .SAV      59  27-Feb-90
 13 Files,  240 Blocks
 518 Free blocks
```

При открытии файла производится поиск его имени в каталоге диска и определение номера начального блока. По завершении операции программа поиска файла в каталоге больше не нужна и попусту занимает место в памяти. Операции чтения и записи производятся относительно начального блока файла. Программа пользователя передает операционной системе информацию о номере блока (относительно начала файла), с которого начинается обмен информацией, количество слов обмена, а также адрес, куда поместить или откуда взять данные из памяти. Далее *RMON* вычисляет номер блока на диске, т. е. к заданному номеру блока прибавляет начальный и передает информацию драйверу.

При такой организации работы программа поиска файла и определения его местонахождения используется всего один раз. Поэтому она выделена в отдельную подпрограмму *USER* (*USER* хранится в одном файле вместе с монитором) и загружается в память по мере необходимости.

Процесс загрузки происходит следующим образом. После получения запроса от программы на открытие файла операционная система пытается разместить в неиспользуемую область памяти *USER*. Если свободная область мала, то производится свопинг, т. е. часть программы пользователя переносится на диск в файл *SWAP.SYS*, а на высвобожденное место загружается *USER*. После выполнения необходимых действий программа (из файла *SWAP.SYS*) восстанавливается. Таким образом, для ускорения работы системы требуется, чтобы файл *SWAP.SYS* — область свопинга — и файл монитора операционной системы (например, *RT11CD.SYS*) находились в начале диска.

Первая строка сообщает о текущей дате (25 декабря 1991 г.). Затем выводится информация о файлах, находящихся на диске, их размерах и дате создания. Дата выводится в формате: день — месяц — год.

Месяц выводится сокращением английского названия: *Jan* — январь, *Feb* — февраль. Например, запись

```
RT11CD.SYS    70      23-Aug-86
```

информирует о том, что файл *RT11CD.SYS* имеет размер 70 блоков, создан 23 августа 1986 г.

Последние две строки информируют о том, что всего на диске 13 файлов, занимают они 268 блоков и имеются еще 518 свободных блоков.

Для получения информации о файлах, находящихся на диске, установленном в первом кармане дисковода, после команды *DIR* следует набрать имя устройства *MZ1:*.

```
MZ0>DIR MZ1:
```


Внимание! Не забывайте после набора команды нажимать на клавишу ⟨ВВОД⟩.

После введения команды произойдет обращение к нижнему карману дисковода (загорится световой индикатор выбора первого кармана дисковода) и на экран дисплея будет выведен каталог диска. Например:

```
 25-Dec-91
BASIC .BIN    54P 10-Feb-87     BASIC .COM     1  11-Aug-90
NET   .COM     1  11-Aug-90     UDAN  .COD     6  01-Jan-91
UDAN  .ABC     6  01-Jan-91     TIKTAK.ABC    3P  15-Jan-90
  6 Files,   71 Blocks
715 Free blocks
```

Сообщение

```
BASIC .BIN     54P      10-Feb-87
```

информирует о том, что файл *BASIC.BIN* создан 10 февраля 1987 г. и его размер 54 блока. Символ *P* после размера указывает на то, что файл защищен от удаления.

Если набрать предыдущую команду без двоеточия после *MZ1*, то результат будет следующим:

```
MZ0>DIR MZ1

25-Dec-91

 0 Files,    0 Blocks
518 Free blocks
```

В данном случае ОС сообщила, что на диске, установленном в нулевом приводе, файлов с именем *MZ1* нет.

Внимание! Двоеточие — признак внешнего устройства.

Имя внешнего устройства состоит из двух символов и цифры (необязательной). Цифра указывает на номер устройства. Если номер в имени опущен, то он принимается за 0. Например:

MZ0: — дисковод 0 карман.

MZ1: — дисковод 1 карман.

MZ0: — дисковод 0 карман.

LP: — устройство печати (так как к ЭВМ подключается одно устройство печати, то номер игнорируется. Обращения к *LP*: или к *LP1*: приведут к одному результату).

MZ — это файл с именем *MZ* (так как после имени не стоит двоеточие).

LP0 — это файл с именем *LP0*.

Далее приводятся наиболее распространенные имена внешних устройств:

DK: — устройство по умолчанию. После загрузки ОС совпадает с именем и номером устройства, с которого была загружена ОС. В процессе работы можно переназначать. Устройство по умолчанию указывается в приглашении ОС. Например: *MZ0* — устройство по умолчанию *MZ0*, *MZ1*⟩ — устройство по умолчанию *MZ1*:, *RD0*⟩ — устройство по умолчанию «квазидиск» *RD:* (номер «квазидиска» роли не играет, так как «квазидиск» на ЭВМ один (см. ниже). Если перед именем файла не указано устройство, то ОС ищет файл на *DK:*.

SY: — системное устройство. Совпадает с именем и номером устройства, с которого была загружена ОС. В процессе работы не изменяется. Например, если ОС загружена с верхнего кармана дисковода, то обращение к *MZ0*: и к *SY* — одно и то же действие.

MZ: — накопитель на гибком магнитном диске. После имени можно употреблять номер устройства. Как правило, к ЭВМ подключен дисковод с двумя карманами. Верхний имеет номер 0, нижний — номер 1. Всего можно подключить не более 4 карманов.

TT: — дисплей (экран) для вывода и клавиатура для ввода.

LP: — устройство печати (принтер).

RD: — «квазидиск». К УКНЦ можно подключить цветной дисплей, но, как правило, используются монохромные (черно-белые). Поэтому память, предназначенная для хранения цветов, можно использовать для хранения данных.

Загрузка операционной системы производится в два этапа. Сначала производится считывание программы из нулевого блока диска, называемой первичным загрузчиком и содержащей в себе подпрограмму чтения с диска, и передач управления. Затем первичный загрузчик считывает вторичный загрузчик, размещаемый на диске со второго по пятый блок, и передает ему управление. Вторичный загрузчик, используя подпрограмму чтения первичного загрузчика, находит на диске файл монитора и загружает его. Первичный загрузчик находится в драйвере системного устройства, а вторичный — в файле монитора операционной системы.

NL: — фиктивное устройство. Данные, выводимые на устройство *HL:* будут выводиться в пустоту. Применяется для совместимости программного обеспечения.

PP: — ОЗУ периферийного процессора.

Внешние устройства делятся на два типа: устройства прямого и последовательного доступа. Дисковод — это устройство прямого доступа. Мы можем найти на нем запись сразу, перейдя на нужную дорожку. Устройство печати *LP:* — последовательного доступа. Для вывода последнего символа в строке надо сначала вывести все предыдущие. Это относится и к устройству *TT:* для ввода с клавиатуры и вывода на экран.

После получения команды ***DIR*** операционная система считывает с устройства блок, где хранится информация о файлах (для устройств прямого доступа это 6-й блок). Поэтому команда ***DIR MZ1:*** имеет смысл, а ***DIR LP:*** нет, так как невозможно найти на печатающем устройстве 6-й блок (вышесказанное относится и к *TT:*). Таким образом, команда ***DIR*** будет выполняться корректно для следующих устройств: *SY:*, *DK:*, *MZ:*, *RD:*. Устройство *PP:* тоже прямого доступа, но не обладает файловой структурой.

С командой ***DIR*** можно употреблять следующие ключи:

1. ***/BLOCKS*** (***/BLO***) — распечатать каталог с указанием номеров блоков, с которых начинается файл.

```
MZ0>DIR/BLO MZ1:

 25-Dec-91
BASIC .BIN  54P 10-Feb-87  14     BASIC .COM   1  11-Aug-90  68
NET   .COM   1  11-Aug-90  69     UDAN  .COD   6  01-Jan-91  74
UDAN  .ABC   6  01-Jan-91  80     TIKTAK.ASC  3P  15-Jan-90  86
  6 Files,   71 Blocks
715 Free blocks
```

Файл *BASIC.BIN* имеет размер 54 блока (*54P*), защищен от удаления, создан 10 февраля 1987 г. и располагается на диске, начиная с блока 14. Файлы занимают последовательно расположенные блоки. Первый свободный блок после *BASIC.BIN* соответственно будет 14+54=68. Следующий файл *BASIC.COM* как раз и размещается на диске, начиная с блока 68. Файл *NET.COM* располагается с блока 69 и имеет размер 1 блок. Следовательно, следующий файл может начинаться с 69+1=70 блока, но реально он располагается с 74. Значит,4 блока: 70, 71, 72, 73 — свободные. Для выдачи информации о свободных блоках используется следующий ключ.

2. ***/FULL*** (***/FU***) — распечатать полный каталог.

```
MZ0>DIR/BLO/FU  MZ1:

 25-Dec-91
BASIC .BIN  54P 10-Feb-87  14     BASIC .COM   1  11-Aug-90  68
NET   .COM   1  11-Aug-90  69     < Unused >   4              70
UDAN  .COD   6  01-Jan-91  74     UDAN  .ABC   6  01-Jan-91  80
TIKTAK.ASC  3P  15-Jan-90  86     < Unused > 711              89
  6 Files,   71 Blocks
715 Free blocks
```

Информация ⟨*Unised*⟩ сообщает о том, что в этом месте находятся свободные блоки:

⟨*Unised*⟩ 4 70 — свободно 4 блока, начиная с 70;

⟨*Unised*⟩ 711 89 — свободно 711 блоков, начиная с 89.

Таким образом, на диске свободных 4+711=715 блоков, о чем и информирует строка:

715 *Free blocks.*

3. ***/Print*** (***/PRI***) — вывести каталог на устройство печати.

Для использования этого ключа надо включить принтер и набрать команду ***DIR/PRI***. При этом информация о каталоге диска будет выведена на печатающее устройство.

MZ0⟩DIR/PRI — вывести каталог на печать.

MZ0⟩DIR/PRI/FU/BLO SY — вывести каталог системного диска на печать с указанием свободных областей и номеров блоков.

4. ***/ALPHABETIZE*** (***/ALP***) — распечатать каталог диска в алфавитном порядке.

При указании этого ключа имена файлов выводятся не так, как они (файлы) расположены на диске, а в алфавитном порядке.

5. ***/VOLUMEID*** (***/VOL***) — распечатать индентификатор диска (имя диска) и имя владельца. Эта информация заносится на диск в процессе инициализации.

6. ***/BAD BLOCKS*** (***/BAD***) — проверить диск (или файл) на наличие плохих блоков.

7. ***/EXCLUDE*** (***/EXC***) — вывести информацию о файлах, исключая указанные. Далее приводится пример команды ***DIR*** без ключа ***EXC*** и с ним.

```
MZ0>DIR/VOL

 25-Dec-91
Volume ID: Диск 01
Owner    : Школа 24
SWAP  .SYS    26  02-Sep-86     RT11CD.SYS   78  23-Aug-86
LP    .SYS     2  01-Jun-89     TT    .SYS    2  01-Jan-86
```

При работе пользователь общается с операционной системой посредством ввода команд. Например, при копировании требуется ввести имя команды, имена копируемых файлов и имя устройства, куда будут скопированы файлы. При этом пользователь должен сначала посмотреть каталог диска, запомнить имена нужных файлов, а затем набрать эти имена в команде. Для создания более удобной среды работы используются программы-оболочки. Наиболее распространенной оболочкой является Norton Commander (NS) для *IBM*-совместимой техники. В последнее время подобные оболочки получили распространение и на других видах техники, в том числе и для УКНЦ создана подобная оболочка.

```
MC     .SYS     6  10-Aug-87    MZ     .SYS     4  15-Jan-90
DIR    .SAV    19  10-Aug-87    PIP    .SAV    29  15-Jan-90
DUP    .SAV    45  10-Aug-87    DAY    .SAV     4  31-Mar-87
STARTS.COM      1  01-Sep-90    BASIC  .COM     1  01-Feb-88
BASIC  .SAV    59  27-Feb-90
 13 Files.   260 Blocks
 518 Free blocks

MZ0>DIR/VOL/EXC *.SYS

25-Dec-87
Volume ID: Диск 01
Owner    : Иванов 24
DIR    .SAV    19  10-Aug-87    PIP    .SAV    29  15-Jan-90
DUP    .SAV    45  10-Aug-87    DAY    .SAV     4  31-Mar-87
STARTS.COM      1  01-Sep-90    BASIC  .COM     1  01-Feb-88
BASIC  .SAV    59  27-Feb-90
  8 Files.   158 Blocks
 518 Free blocks
```

Распечатана информация о файлах, за исключением всех имеющих расширение *SYS*. Знак * называется знаком групповой операции. В этом случае ОС считает, что * может заменять любое имя. Например:

**.SYS* — файлы с любыми именами, но обязательно с расширением.

Например: *MZ.SYS, SWAP.SYS.*

*TST.** — файлы с именами *TST* и любым расширением. Данная запись эквивалентна записи *TST*, так как если указано *только имя*, то ОС ищет все файлы с таким именем. Например: *TST.SAV TST.LST, TST.C, TST.COD, TST*. (последний файл не имеет расширения). Запись вида: *TST* — файл (один), имеющий имя *TST* и расширение, состоящее из одних пробелов.

S — файлы, содержащие в своем имени символ *S* и имеющие любое расширение. Место положения символа (символов) *S* — любое. Например: *TST.SAV, TST.LST, SOS.SAV, STOP.COD, DIS.MAC, SOSFS.DOC.*

*I*D* — файлы, имена которых *обязательно* начинаются символом *I* и *обязательно* заканчиваются символом *D*, с любым расширением. Например: *ID.COM, IHHD.SAV.*

*I*D** — файлы, имена которых *обязательно* начинаются символом *I* и их расширение *обязательно* начинается символом *D*. Например: *I, D, IRA.DOC, ICI.DDI.*

Более полную информацию о ключах, применяемых в командах, можно получить в приложении по ОС ФОДОС.

3.2. Копирование файлов

При работе на ЭВМ постоянно приходится перемещать файлы на диске, создавать архивные (запасные на случай порчи основного рабочего файла) копии, обмениваться файлами с другими пользователями и т. д. Для этого применяется команда *COPY* (*COP*). Формат записи команды:

MZ0⟩COPY {/ключ} имя_исходного_файла имя_файла_результата

Например:

```
MZ0>COP MZ0:BASIC.SAV MZ1:
```

Файл с именем *BASIC.SAV*, находящийся на диске, установленном в верхнем кармане дисковода (*MZ0:*), копируется на диск, установленный в нижнем кармане дисковода (*MZ1*). После выполнения этой команды на нижнем диске появится файл *BASIC.SAV*. Этот файл будет точной копией файла *BASIC.SAV* верхнего диска. Файл с *MZ0*: *не удаляется*. Таким образом можно размножать файлы.

Однако если на диске *MZ1*: уже находился файл под именем *BASIC.SAV*, то он будет заменен на новый (только что скопированный) файл. Замена старого файла на новый будет производиться, если старый файл *не защищен* от удаления. В противном случае копирование производиться не будет. Следующая команда:

```
MZ0>COP MZ0:BASIC MZ1:
```

Выполняет копирование всех файлов с именем *BASIC* и любым расширением. Можно скопировать все файлы с одного диска на другой. Для этого после имени устройства не надо указывать имена файлов. Например:

```
MZ0>COP MZ0: MZ1:
```

Копируются все файлы с диска *MZ0*: на диск *MZ1*:. Однако если на диске *MZ0*: находились системные файлы, то они копироваться не будут. Для включения в операцию копирования системных файлов надо указать ключ *SYS*. При описании файлов можно использовать указатели групповых операций. Например:

```
MZ0>COP MZ0:*.LST MZ1:
```

Копирование всех файлов, имеющих расширение *LST*, с диска *MZ0*: на диск *MZ1*:.

Следующая команда копирует *только* системные файлы с диска *MZ0*: на диск *MZ1*:.

```
MZ0>COP MZ0:*.SYS MZ1:
```

Как и команда *DIR*, команда *COPY* позволяет использовать ключи. Некоторые из

После запуска программы-оболочки на экран выводятся каталоги двух дисков, так называемые окна. Пользователь может, перемещая курсор по каталогу, выделять нужные файлы. В нижней строке выводятся списки команд. Например, для копирования с одного диска на другой достаточно выделить в одном окне файлы, при этом их имена отображаются инверсно, затем нажать на клавишу ⟨*M*⟩, и операция копирования на другой диск выполнится. Пользователь может менять устройства, каталоги которых выводятся в окнах. Из оболочки можно запускать программы на выполнение. Для этого надо подвести курсор к выбранному файлу и нажать на ⟨ВВОД⟩. Оболочка позволяет создавать меню пользователя. При нажатии на клавишу ⟨*U*⟩ на экране появляется еще одно окно с меню.

Пакеты программ, описанные в данной статье, можно приобрести в учебно-методическом центре «Информвидео». См. рекламу на странице 70.

них совпадают с ключами команды *DIR*. Далее приводятся наиболее употребляемые ключи команды *COPY*. Часть из них будет более подробно рассмотрена в разделе «Создание системного диска».

1. */SYSTEM* (*/SYS*) — включить в операцию копирования системные файлы, т. е. файлы, имеющие расширение *SYS*. Без этого ключа ОС игнорирует системные файлы. Следующая команда копирует все файлы, включая системные, с диска *MZ0:* на диск *MZ1:*

```
MZ0>COP/SYS MZ0: MZ1:
```

Внимание! Команда типа COP/SYS MZ0:.ASC MZ1: смысла не имеет, так как копируются только те файлы, которые имеют расширение ASC*, и ключ *SYS* игнорируется.

2. */BOOT* (*/BOO*) — копирует системный загрузчик. (Подробное описание в разделе «Создание системного диска».)

3. */EXCLUDE* (*/EXC*) — аналогично ключу команды *DIR*. Например:

```
MZ0>COP/EXC MZ0:*.COD MZ1:
```

Копировать все файлы, кроме тех, которые имеют расширение *COD*, с диска *MZ0:* на диск *MZ1*. (Системные файлы тоже не будут копироваться, так как нет ключа *SYS*.)

4. */TGNORE* (*/TGN*) — игнорировать ошибки при копировании. Если ключ не используется, а копируемый файл запорчен (например, из-за плохого диска), то ОС выдаст сообщение об ошибке и завершит операцию аварийно (файл скопирован не будет). При использовании этого ключа копирование запорченного файла продолжится. Таким образом можно попытаться восстановить запорченный файл.

5. */REPLACE* (*/REP*) — копирование с заменой. Этот ключ можно не указывать, так как ОС использует его при копировании по умолчанию (если не указан ключ */NOREP*).

6. */NOREPLAGE* (*/NOREP*) — копирование без замены. При копировании ОС заменяет старые файлы на новые (если они не защищены от удаления). Данный ключ запрещает копирование, если на выходном устройстве обнаружен файл с таким же именем. Например:

```
MZ0>COP/NOREP MZ0:TEST MZ1:
```

Копируются файлы с именем *TEST* (расширение любое) с диска *MZ0:* на диск *MZ1:*. Пусть на диске *MZ0:* находятся следующие файлы с именем *TEST : TEST.SAV*, *TEST.ASC*, *TEST.LST*, а на диске *MZ1:* имеются следующие файлы с именем *TEST : TEST.ASC*, *TEST.LST*. Тогда при выполнении операции будет скопирован только файл *TEST.SAV*, а файлы *TEST.ASC* и *TEST.LST* копироваться не будут.

7. */NOQUERY* (*/NOQ*) — копирование без запроса. Его можно не указывать, так как ОС принимает этот ключ по умолчанию.

8. */QUERY* (*/Q*) — копирование с запросом. При использовании этого ключа ОС будет требовать подтверждения включения каждого файла в операцию. (Данный ключ применяется во многих командах ОС.) На каждый запрос ОС пользователь должен ответить *Y* и нажать на ⟨ВВОД⟩, если файл надо копировать, или *N* — если не надо. (*N* можно не набирать, а сразу нажимать на ⟨ВВОД⟩.) Например:

```
MZ0>COP/Q MZ0:TEST MZ1:
```

После этого ОС выведет сообщения и задаст вопрос (знак?):

```
        Files copied
MZ0:TEST.SAV      to MZ1:TEST.SAV ?Y
MZ0:TEST.BIN      to MZ1:TEST.BIN ?N
MZ0:TEST.LST      to MZ1:TEST.LST ?
MZ0:TEST.ASC      to MZ1:TEST.ASC ?Y
```

В результате файлы:

TEST.SAV — будет скопирован (мы ответили *Y*).

TEST.BIN — не будет скопирован (ответ *N*).

TEST.LST — не будет скопирован (нажали просто на ⟨ВВОД⟩, что аналогично ответу *N*).

TEST.ASC — будет скопирован (ответ *Y*).

9. */NEWFILES* (*/NEW*) — копировать файлы, имеющие текущую дату создания (т. е. те файлы, которые созданы сегодня).

10. */DELETE* (*/DEL*) — копирование с удалением исходного файла. Например:

```
MZ0>COP/DEL MZ0:TEST.DOC MZ1:
```

После завершения операции копирования исходный файл *TEST.DOC*, находящийся на диске *MZ0:* будет удален. Удаляются только незащищенные файлы.

11. */WAIT* — ожидать подтверждения установки необходимого диска.

Когда ОС принимает команды пользователя, то она загружает из файла монитора необходимые программы. Таким образом, при выполнении команд *нельзя* убирать системный диск. Ключ *WAIT* позволяет это обойти. Допустим, файл *TEST.DOC* находится на рабочем диске (диск # 1). Его надо скопировать на другой рабочий диск (диск # 2). Для этого требуется ввести команду:

```
MZ0>COP/WAIT MZ0:TEST.DOC MZ1:
```

ОС, встретив в команде ключ *WAIT*, загрузит необходимые ей программы, но операцию копирования выполнять не будет, а потребует от пользователя установки входного диска:

```
Mount input volume  in MZ0:
                        Continue ? Y
```

Необходимо удалить из нулевого кармана системный диск и поставить вместо него диск # 1. После этого надо ответить *Y*.

Затем ОС выдаст запрос на установку диска, на который будет производиться копирование:

```
Mount output volume in MZ1:;
                          Continue ? Y
```

Следует установить выходной диск и ответить *Y*.

После завершения операции копирования ОС потребует установки системного диска:

```
Mount system volume in MZ0:;
                          Continue ? Y
```

Необходимо установить системный диск и ответить *Y*.

В команде *COPY* можно перечислять файлы, включаемые в операцию. Одновременно можно указывать не более 6 файлов. Например: `MZ0>COP TEST.DOC,BASIC MZ1:`

Копировать файл *TEST.DOC* и все файлы с именем *BASIC* (расширение любое) на диск *MZ1*:

`MZ0>COP MZ1:TEST.DOC,MZ1:BASIC MZ0:`

Копировать файл *TEST.DOC* и все файлы с именем *BASIC* (расширение любое) с диска *MZ1*: на диск *MZ0:*. Эту команду можно записать по-другому:

`MZ0>COP MZ1:(TEST.DOC,BASIC) MZ0:`

Скобки указывают, что файлы надо брать с одного устройства *MZ1:*.

Число реально копируемых файлов может превышать 6 (например, файлов с именем *BASIC* и любым расширением может быть сколько угодно: *BASIC.SAV*, *BASIC.BIN* и т. д.), но в команде нельзя указывать более 6 имен.

Продолжение следует

Л. ЛЕБЕДЕВ

Программы для УКНЦ: подход АТТИКа

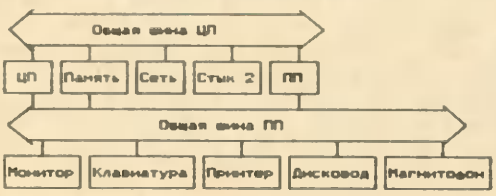
УКНЦ — учебный компьютер «Научного центра» — является сегодня одним из самых массовых школьных компьютеров. В течение нескольких лет группой АТТИК лаборатории вычислительных методов механико-математического факультета МГУ ведется разработка программного обеспечения для КУВТ УКНЦ. За это время был накоплен большой опыт и сформировались определенные взгляды на структуру и организацию программных систем, максимально использующих ресурсы этого компьютера. Расскажем об одном из использованных на практике подходов.

Но сначала — несколько слов об особенностях архитектуры УКНЦ, которые не только важны для понимания последующего, но и интересны сами по себе.

Организация УКНЦ, взаимодействие ЦП и ПП

На приведенном рисунке показаны основные компоненты УКНЦ: устройство ввода (клавиатура), устройства вывода (монитор, принтер), память, внешние запоминающие устройства (дисковод, магнитофон), устройства связи с внешним миром (сеть, интерфейс «стык С2»). Своеобразие этой схемы заключается в наличии двух процессоров: центрального (ЦП) и периферийного (ПП). Комплекс из периферийного процессора, его памяти и внешних устройств (ВУ) сам по себе уже является полноценным персональным компьютером: зачем же разработчикам понадобилось изменять традиционную организацию ЭВМ?

1



Это объясняется желанием освободить ЦП от управления ВУ и отдать все его ресурсы прикладной программе. Кроме того, введение посредника упрощает взаимодействие ЦП и ВУ: прикладная программа, выполняемая ЦП, может формировать запросы к устройствам в абстрактной форме, а ПП преобразует их в конкретные команды управления ВУ.

ЦП и ПП представляют собой два одинаковых 16-разрядных процессора КМ1801ВМ2. Термин «16-разрядный» означает, что единица информации, над которой производится большинство операций, состоит из 16 двоичных разрядов. Система команд процессора КМ1801ВМ2 во многом аналогична используемым в таких ЭВМ, как

PDP-11/70 фирмы DEC, а также ЭВМ отечественного производства СМ-4, «Электроника 79», БК-0010, ДВК, систематическое изложение набора инструкций которых можно найти в [1], [2], а также в цикле статей Ю. Зальцмана в «ИНФО».

Характерной особенностью УКНЦ является наличие двух магистралей передачи данных: «Общая шина ЦП» и «Общая шина ПП». Логически общая шина (ОШ) — это совокупность правил адресации (оперативной памяти, регистров процессора и внешних устройств) и правил взаимодействия (перечисленных компонентов). Эти правила реализуются специальными схемами процессора, памяти и контроллерами ВУ. Физически ОШ — многопроводная линия, проходящая последовательно через процессор, память и ВУ. Обмен данными между любыми двумя устройствами выполняется через ОШ, причем используется единая система адресации для слов памяти и регистров внешних устройств, т. е. процессор может рассматривать регистры ВУ как ячейки ОЗУ и обращаться к ним с помощью адресных команд. Регистры состояния и данных клавиатуры, например, имеют в магистрали ПП адреса 177700 и 177702 соответственно.

Но как же ЦП взаимодействует с устройствами, подключенными к общей шине ПП?

Каждый из процессоров подключен к магистрали своего «коллеги» и рассматривается им как внешнее устройство; совокупность регистров последнего называется каналом. Таких каналов три: два двунаправленных (от ЦП к ПП и обратно) и один однонаправленный (только от ЦП к ПП). В табл. 1 приведены адреса регистров каналов в магистралях ЦП и ПП.

Таблица 1

	Магистраль ЦП				Магистраль ПП			
	РСП	РДП	РСИ	РДИ	РСП	РСИ	РДП	РДИ
Канал 0	177560	177562	177564	177566	177860	177870	177862	177870
Канал 1	176660	176662	176664	176666	177866	177876	177862	177872
Канал 2	нет	нет	176674	176676	177866	нет	177864	нет

Здесь РСП (РСИ) – регистр состояния приемника (источника);
РДП (РДИ) – регистр данных приемника (источника).

Используя каналы, ЦП взаимодействует с такими устройствами, как монитор, клавиатура, дисковод, магнитофон, принтер. Нулевой канал выделен для ввода символов с клавиатуры и вывода информации на экран, первый — для взаимодействия с принтером, второй — для передачи запросов к остальным устройствам.

Организация памяти УКНЦ. Любая ЭВМ имеет ОЗУ — место, где хранятся данные и программы. В этом смысле в корпусе УКНЦ находятся две ЭВМ, так как каждый ее процессор имеет свое ОЗУ. Объем ОЗУ ЦП — 56 Кбайт, ПП — 32 Кбайт.

Изображение хранится в видеопамяти, разбитой на три плана по 32 Кбайт каждый. Цвет точки на экране кодируется тремя битами, по одному из каждого видеоплана. Октету последовательно расположенных точек соответствуют три байта, идентифицируемые одним на троих адресом в адресном пространстве видеопамяти. При максимальном разрешении 640×288 точек в каждом видеоплане остаются неиспользованными около 8 Кбайт.

Доступ ЦП и ПП к своим ОЗУ осуществляется традиционным образом — командами чтения-записи. «Общение» с видеопамятью организовано иначе.

Видеопланы рассматриваются как внешние устройства и представлены в магистралях ЦП и ПП регистрами адреса и данных (табл. 2). Каждое значение регистров ад-

Таблица 2

Магистраль ЦП			Магистраль ПП			
РА	РД1	РД2	РА	РД0	РД1	РД2
176640	176642	176643	177010	177012	177014	177015

реса (РА) однозначно определяет ячейки памяти, доступ к которым осуществляется через регистры данных (РД1 и РД2 для ЦП, РД0—РД2 для ПП). Это соответствие представлено в табл. 3. Из нее видно, что ОЗУ ЦП и ОЗУ ПП также доступны через эти регистры и могут, таким образом, рассматриваться как внешние устройства; эта особенность используется для обращений ПП к ОЗУ ЦП.

Таблица 3

Магистраль ЦП			
РД2 ЦП	РД1 ЦП	РА ЦП	
ОЗУ ЦП (1)	ОЗУ ЦП (0)	000000	ОЗУ ПП (0)
⋮	⋮	⋮	⋮
ОЗУ ЦП (177777)	ОЗУ ЦП (177776)	077777	ОЗУ ПП (77777)
план2 (1)	план1 (0)	100000	план0 (0)
⋮	⋮	⋮	⋮
план2 (77777)	план1 (77777)	177777	план0 (77777)
РД2 ПП	РД1 ПП	РА ПП	РД0 ПП
Магистраль ПП			

ЦП несколько ущемлен в своих правах: ему недоступны ОЗУ ПП и нулевой видеоплан. Это отчасти компенсируется наличием каналов связи между ЦП и ПП.

Режимы работы ЦП. Центральный процессор УКНЦ может работать в двух режимах: USER и HALT. В режиме USER адресное пространство ЦП состоит из ОЗУ (диапазон адресов от 0 до 157777) и страницы ввода-вывода, где расположены регистры устройств (адреса от 160000 до 177777). Прикладной программе доступны только 56 Кбайт памяти.

В режиме HALT страница ввода-вывода отключена, с адреса 160000 и выше располагается системная область ОЗУ. В этом режиме работы процессору доступны все 64 Кбайт (ОЗУ плюс системная область), а для обращения к регистрам устройств используются специальные команды.

USER является общеупотребительным, используется операционной системой RT-11 и всеми прикладными программами. Режим HALT — специальный. Его использует пультовой монитор, располагающийся в системной области. Кроме него, системная область содержит программную реализацию арифметических операций с вещественными числами.

HALT можно рассматривать как резерв на случай катастрофической нехватки памяти.

Программное обеспечение базового уровня. ПЗУ, доступ к которому имеет только периферийный процессор, содержит начальные загрузчики с внешних устройств, пультовый монитор, программы тестирования, управления внешними устройствами, эмуляторы терминалов VT52 фирмы DEC и «Электроника 15ИЭ-00-013» отечественного производства.

Терминал — это устройство ввода-вывода эпохи мини-ЭВМ, состоящее из дисплея и клавиатуры. При нажатии на клавишу терминал посылал в компьютер определенные коды, а вывод информации на экран управлялся со стороны компьютера специальными командами. Эмулятор же терминала — это исполнитель, умеющий расшифровывать командные последовательности терминала. Эмулятор «ведет себя» так, что прикладная программа не подозревает, что взаимодействует не с реальным устройством, а с его программной имитацией. Именно с эмулятором терминала взаимодействует ЦП, когда занимается вводом с клавиатуры и выводом на экран.

Это сделано для того, чтобы обеспечить ДВК-совместимость УКНЦ и перенести на УКНЦ множество разработанных ранее программ, в частности операционную систему RT-11. Однако если возможности, предоставляемые терминалами, кого-то не удовлетворяют, можно загрузить в ОЗУ ПП и запустить свою программу, которая возьмет на себя их функции полностью или частично.

Сетевые операционные среды для КУВТ УКНЦ

КУВТ УКНЦ состоит из рабочего места преподавателя (РМП — машина с дисководом) и рабочих мест учеников (РМУ — бездисковые машины). На РМУ информация хранится в ОЗУ и пропадает после выключения питания. Таким образом, полноценной ЭВМ в КУВТ является только РМП, на ученические машины программы и файлы, необходимые для работы, нужно пересылать по сети.

Систему, позволяющую ученикам запускать программы и обмениваться информацией с диском учителя, назовем сетевой операционной средой — СС. В настоящее время сложились общие принципы организации операционных сред, которые в той или иной мере прослеживаются в большинстве изделий подобного рода.

Традиционный подход. СС с точки зрения сторонников традиционного подхода представляет собой сетевую операционную систему. На РМУ запускается командный монитор ОС RT-11, после чего взаимодействие с ученическим и учительским компьютерами внешне ничем не различается; чтобы посмотреть список файлов, ученик может воспользоваться командой DIR, для запуска программ — командой DUN и т. д. Для доступа к дискам в такой ОС вводится новое внешнее устройство, например VD (виртуальный диск), роль которого выполняет аппаратно-программный комплекс: сеть, сетевой монитор на РМП, дисковод. Диску MZ0 при этом соответствует устройство VD0 а MZ1 — VD1.

Каждое ВУ в системе RT-11 имеет свой драйвер, специальную программу-посредника между RT-11 и устройством. Драйвер хранится на диске в файле ‹имя устройства›.SYS (например, LP.SYS — драйвер принтера, MZ.SYS — дисковода) и может быть загружен в память по команде LOAD ‹имя устройства›. Обращение к ВУ оформляется в виде системных запросов, которые RT-11 переадресует соответствующему драйверу, а тот в свою очередь через регистры ВУ и магистрали процессора получает доступ к устройству.

Драйвер виртуальных дисков преобразовывает системные запросы в сообщения, передаваемые по сети на РМП, где сетевой монитор обрабатывает их в порядке поступления и взаимодействует через драйвер MZ.SYS с реальными дисками.

Сетевая операционная система обладает по крайней мере одним неоспоримым преи-

муществом: любая программа, работающая на РМП, может быть запущена на РМУ без каких бы то ни было переделок. Однако при существующей низкой скорости передачи данных по сети запуск на двух-трех РМУ программ, ведущих пусть даже не очень интенсивный обмен с дисками РМП, значительно увеличивает время реакции системы на действие учеников. Для ускорения работы используется электронный диск на РМУ. Скопировав на него все необходимые прикладной программе файлы, можно сильно сократить обмены с дисками учителя.

Электронный диск подключается в качестве ВУ к системе RT-11 и размещается на двух видеопланах, к которым имеет доступ ЦП, для вывода же на экран используется только нулевой видеоплан. Поскольку цветные мониторы в школы не поставляются, происходит не потеря цвета, а потеря возможных градаций яркости, что не такая уж большая плата за 64-Кбайтовое электронное дисковое пространство на бездисковых РМП.

Сетевая операционная система с электронным диском уже вполне приемлема для использования в классе УКНЦ. Она позволяет запускать на РМУ даже оверлейные программы (оверлейной называется программа, организованная так, что некоторые ее более или менее независимые части (оверлеи) во время выполнения подгружаются с диска в одну и ту же область ОЗУ, что сокращает объем памяти, необходимый для работы программы). Конечно, такую программу можно запустить и в сетевой операционной системе без электронного диска, но тогда загрузка оверлеев потребует интенсивного обмена с РМП, что для интерактивных программ неприемлемо.

Основной недостаток традиционного подхода — довольно жесткие ограничения, налагаемые им на объем программ, запускаемых на РМУ. Эти ограничения связаны с распределением памяти в системе RT-11.

Распределение памяти в ОС RT-11. RT-11 и ее модификации разрабатывались в расчете на однопроцессорные компьютеры. Поэтому адаптированные к УКНЦ версии этой системы используют периферийный процессор только как эмулятор внешних устройств и его ОЗУ никак не учитывается при распределении памяти. И система, и прикладная программа располагаются в ОЗУ ЦП объемом 56 Кбайт.

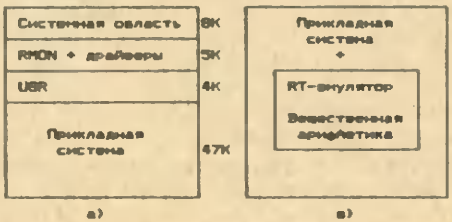
Верхнюю область ОЗУ занимают резидентная (постоянно находящаяся в памяти) часть RT-11 (RMON) и драйверы внешних устройств (рис. 2, а). Ее объем зависит от модификации системы и количества загруженных драйверов (около 5 Кбайт). Кроме резидентной части RT-11 имеет свопируемую часть USR (примерно 4 Кбайт), расположением которой можно управлять. Команда SET USR NOSWAP фиксирует USR в памяти непосредственно перед RMON. Команда SET USR SWAP разрешает прикладной программе самой определить, куда при необходимости будет загружаться USR; при этом USR может покрыть собой какой-то кусок программы. В таком случае перед загрузкой USR система RT-11 организует свопинг: покрываемая часть прикладной программы копируется на диск в файл SWAP.SYS, а USR занимает ее место. Когда USR больше не нужен, сохраненная в SWAP.SYS информация возвращается назад в память. Область ОЗУ, не занятая резидентной и свопируемой частями системы, отдается в распоряжение прикладной программе (это примерно 47 Кбайт, если USR зафиксирован в памяти, и 51 Кбайт — в противном случае).

Отметим, что в сетевой операционной среде невозможно организовать корректный свопинг нескольких задач с РМУ на диск учителя, так как RT-11 не поддерживает такую возможность. Поэтому в памяти РМУ USR должен быть зафиксирован, следовательно, объем неоверлейной программы ограничен 47 Кбайт. Размер оверлейных программ ограничивается размером электронного диска (64 Кбайт). При этом надо иметь в виду, что кроме самой программы на диске могут размещаться и другие файлы.

Эти ограничения, присущие сетевой операционной системе, не позволяют воспользоваться ею для организации работ в классе УКНЦ с языковыми средами типа Фортран-Мира [3] и КуМира [4]. Чтобы вписаться в рамки адресного пространства ЦП, программы должны быть оверлейными, а нужное им дисковое пространство значительно превышает возможности электронного диска. Пришлось искать новые решения, и поиск увенчался успехом.

Нетрадиционный подход. При критическом анализе традиционной схемы явно вырисовываются три направления ее модернизации: сокращение пространства, занимае-

Распределение памяти на РМУ 2



мого системными компонентами; полное использование ресурсов ПП; подключение системной области ОЗУ ЦП.

RMON и USR, занимающие около 9 Кбайт, содержат обработку системных вызовов. Однако не все эти вызовы реально используются, большинство лежит мертвым грузом. Хорошо бы заменить громоздкую РТ-11 компактным RT-эмулятором, способным обрабатывать ограниченный набор необходимых системных вызовов. Проигрыш в универсальности (не каждая программа сможет работать с RT-эмулятором) будет компенсирован выигрышем нескольких килобайт памяти.

Надежность и пропускная способность сети таковы, что использование ее сразу несколькими интерактивными прикладными программами нереально, поэтому из RT-эмулятора можно исключить ту часть системы, которая занимается обменами с дисками РМУ, постулировав, что во время работы прикладная программа общается только с электронным диском. Файловым обменом может заниматься СС в промежутках между запусками программ, когда имеется достаточный резерв памяти.

Чрезвычайно выгодным оказалось использование потенциала ПП. Он имеет свое ОЗУ, где можно запустить программу. Почему бы не переложить на ПП часть забот ЦП? Во-первых, ПП можно поручить обработку системных вызовов. Благодаря этому RT-эмулятор станет еще более компактным; кроме того, появится возможность увеличить объем электронного диска на 8 Кбайт, не используемых для вывода информации на экран в нулевом видеоплане, не доступном ЦП. Во-вторых, так как клавиатура подключена непосредственно к ПП, ему можно передать предварительную обработку кодов нажатых клавиш. Обычно прикладная программа преобразует исходные коды клавиш в свои внутренние (виртуальные) коды. Поскольку «Миры» используют единые виртуальные коды, то их формирование целесообразно перенести из ЦП в ПП.

Еще один резерв — совершенствование механизма оверлеев. Корень программы (неоверлейная ее часть, постоянно находящаяся в памяти ЦП) хранится еще и на электронном диске. Чтобы устранить такую неоптимальность, приходится выйти за рамки традиционной системы RT-11 и изменить внутреннюю структуру файлов, введя возможность использовать «дырявые» файлы. Логически «дырявый» файл ничем не отличается от обычного и представляет собой последовательность блоков. Его можно копировать, удалять, переименовывать. Однако некоторые его логические блоки физически не занимают места на диске. В виде таких «дырявых» файлов удобно размещать на электронном диске оверлейные программы: корень программы находится в памяти и, следовательно, дисковое пространство, необходимое прежде его копии, можно использовать для других нужд.

Таблица 4

Размер (в Кбайт)	Традиционный подход		Нетрадиционный подход	
	Ф.Мир	КуМир	Ф.Мир	КуМир
адресного пространства ЦП	56		64	
электронного диска	64		72	
памяти, доступной прикладной системе	47		64	
SAV-файла прикладной системы	46	74	44	72
программы ученика, максимальный	13	-	32	16
пространства для ученических файлов на электронном диске	18	-	68	32

Использование режима HALT позволяет отдать прикладной программе системную область ОЗУ. Она содержит программную реализацию арифметических операций с вещественными числами и пультовый монитор; можно безболезненно отказаться от пультового монитора, а операции вещественной арифметики включить в прикладные программы. В результате программа пользователя получит в свое распоряжение почти 64 Кбайт памяти (исключая примерно 2 Кбайт, необходимых RT-эмулятору и арифметике вещественных чисел (рис. 2, б).

Использование изложенных идей при реализации СС, позволяющей организовать работу в классе УКНЦ с системами ФортранМир и КуМир, дало очень хорошие результаты; они приведены в табл. 4.

Литература

1. *Сингер М.* Мини-ЭВМ PDP-11: Программирование на языке ассемблера и организация машины. М.: Мир, 1984.

2. *Лин В.* PDP-11 и VAX-11. Архитектура ЭВМ и программирование на языке ассемблера. М.: Радио и связь, 1989.

3. *Борисенко В. В., Варсанофьев Д. В., Дымченко А. Г.* Ф-практикум // Микропроцессорные средства и системы. 1989. № 6.

4. *Эпиктетов М. Г.* Учебно-производственная среда на основе школьного алгоритмического языка // Материалы V Всесоюзного семинара «Разработка и применение программных средств ПЭВМ в учебном процессе». М., 1989.

Н. ПОДОЛЬСКАЯ

УКНЦ: повышение производительности труда программиста

Чтобы наиболее полно использовать ресурсы компьютера, нужно уметь писать программы минимальных размеров с максимальным быстродействием. Вопрос выбора языка программирования перед профессионалами не стоит — только ассемблер, несмотря на его недостатки. К тому же их можно преодолеть; один из способов приблизить ассемблер к языкам высокого уровня — макрогенерация.

В ассемблере MACRO-11 имеется аппарат текстовых подстановок — макрогенератор. Описание текстовой подстановки в программе называется макроопределением, вызов — макровызовом.

Макроопределение начинается заголовком — директивой. MACRO, первым аргументом которой является имя макроопределения, остальными аргументами — его формальные параметры. Вслед за заголовком идет тело макроопределения: последовательность команд, которую будет представлять в программе макровызов. Конец макроопределения задается директивой.

Макровызовом называется строка, имеющая в поле оператора имя макроопределения, описанного выше этой строки. В поле операндов строки макровызова задаются фактические параметры макровызова.

Макрогенерация выполняется на первом проходе ассемблирования и заключается в том, что макропределения удаляются, а строки макровызова заменяются на тела соответствующих макроопределений; ссылки на формальные параметры в теле макроопределения заменяются ссылками на фактические параметры макровызова.

Например, фрагмент программы

```
.MACRO  ADD2    X
  INC   X
  INC   X
.ENDM
ADD2    R0
```

после первого прохода ассемблирования будет выглядеть так:

```
INC     R0
INC     R0
```

Часто используемые макроопределения можно помещать в отдельные файлы (пакеты), подключаемые к программам при ассемблировании.

На механико-математическом факультете МГУ используется пакет «Макро-85», включающий около 30 макроопределений:

исполнитель (аналогично «package» в языке АДА);

процедура с параметрами и сохранением указанных регистров;

вызов процедур с параметрами и сохранением указанных регистров;

если-то-иначе со сравнениями и логическими операциями;

циклы «пока» и арифметический;

выбор;

операции со стеком и другие сервисные операции;

отладочные возможности (только при программировании на PDP-11 (СМ-4) с применением кросс-технологии).

Использование «Макро-85» качественно меняет характер труда разработчика программ, освобождая его от необходимости следить за метками и состоянием стека, терпеливо набирать массу похожих друг на друга строк и т. п.

Нелегко написать и отладить приводимую ниже программу, еще труднее ее читать.

```
        MOV     R0,     -(SP)
        MOV     LLINE,  R0
        MOV     #LINE,  R1
        MOV     #BUF,   R2
        JSR     %7,     COPY
        MOV     (SP)+,  R0
        ...
COPY:   MOV     R0,     -(SP)
        MOV     R1,     -(SP)
        MOV     R2,     -(SP)
        TST     R0
        BLE     1$
2$:     MOVB    (R1)+,  (R2)+
        SOB     R0,     2$
1$:     MOV     (SP)+,  R2
        MOV     (SP)+,  R1
        MOV     (SP)+,  R0
        RTS     PC
```

При использовании пакета «Макро-85» она же выглядит так:

```
CALL COPY <R0=LLINE,R1=#LINE,
                        R2=#BUF>,SAVE=R0
...
PROC COPY <LEN=R0,LIN1=R1,LIN2=R2>,
                        SAVE=<R0,R1,R2>
  REPEAT LEN
    MOVB (LIN1)+,(LIN2)+
  ENDREP
ENDPRO
```

Поскольку приведенный пример может показаться искусственным, продемонстрируем фрагмент исходного текста редактора-макрогенератора (о нем будет рассказано ниже), реализованного на MACRO-11 с

```
...
PROC    MGTEXT              ;пр."встать на текстовую строку"
  CLR   ERR                 ;есть_ошибка:=нет
  IF    <P EQ #0>           ;если указатель=0 то
    CALL STEMP              ;  очистить стек
    CALL MGINEX             ;  начать расширение макро
    BCS  99$                ;  если ошибка - goto 99
  ENDIF                     ;конец если
  LOOP                      ;цикл
    MOVB @P,R0              ;. код:=@указатель
  WHILE <BYTE R0 NE #$TEXT>     ; пока код != текстовая_строка
    SELECT                  ;. выбор (код)
      CASE <BYTE R0 EQ #_END>  ;при код=конец_макро
        CALL MGBACK         ;. . выйти из макро
        BCS  99$            ;. .
        CALL QSTEMP         ;. . если оно было одно (т.е.стек пуст)
        IF <YES>            ;. . . то
          CLC               ;. . . ответ:=нет текстовой строки
          RETURN            ;. . . выйти из процедуры
        ENDIF               ;. . конец если
        CALL MGMOVP         ;. . передвинуть указатель
      ENDCAS                ;. конец при
      ...                   ;
    ENDSEL                  ;. конец выбора
    IF    <ERROR>           ;. если ошибка то
99$:  MOV #2,ERR            ;. . есть_ошибка:=да
      CLR P                 ;. . указатель:=0 (CARRY:=0)
      RETURN                ;. . выйти из процедуры
    ENDIF                   ;. конец если
  ENDLOO                    ;конец цикла
  SEC                       ;ответ:=да
ENDPRO                      ;конец процедуры
...
```

использованием пакета «Макро-85». Команды ассемблера (CLR, BCS, MOV, CLC, SEC) вполне вписываются в структуру программы, настолько наглядную, что комментарии к управляющим конструкциям кажутся излишними.

Редактор-макрогенератор. Использование пакета макроопределений, подобного «Макро-85», обеспечивает компактные, структурированные тексты и удобный сервис, но существенно замедляет компиляцию. Этот недостаток можно преодолеть, если «скрестить» редактор текстов программ с макрогенератором и использовать для макрогенерации время, в течение которого редактор ожидает действий пользователя, создающего программу. Такой редактор-макрогенератор во время редактирования незаметно для пользователя сам обрабатывает макровызовы, подготовив их для записи в файл: превратит вызовы в комментарии, а рядом поместит их расширения. При подаче программы из файла на экран будут проделаны обратные действия. Например, текст на экране:

```
LOOP                                ;
  CALL    AB  <R1=R0>,SAVE=R1       ;
  WHILE   <R0 NE #0>                ;
ENDLOO                              ;
```

Соответствующий фрагмент файла:

```
;LOOP
10001$:
;
;CALL AB <R1=R0>,SAVE=R1
MOV R1,-(SP)
MOV R0,R1
JSR %7,AB
MOV (SP)+,R1
;
;WHILE <R0 NE #0>
TST R0
BEQ 10002$
;
;ENDLOO
BR 10001$
10002$:
;
```

Теперь сразу видно, что это вызов процедуры COPY (здесь и CALL, и PROC — тоже макро) с параметрами LLINE, # LINE и # BUF, передающимися через регистры R0, R1 и R2 соответственно. При выполнении вызова сохраняется содержимое регистра R0. Ясно также, что процедура COPY заключается в копировании строки LIN1 длины LEN в строку LIN2, длина LEN поступает в регистре R0, адреса строк LIN1 и LIN2 — в R1 и R2 соответственно. При выполнении процедуры сохраняется содержимое регистров R0, R1 и R2. Ну а результатом работы программы будет копирование строки LINE длины LLINE в строку BUF; содержимое регистра R0 не изменится, в R1 будет содержаться адрес LINE, в R2 — адрес BUF.

Использование подобного редактора-макрогенератора (его можно также назвать

редактором-препроцессором) еще больше приближает ассемблер к языкам высокого уровня, причем без особых затрат. Время реакции редактора на действия пользователя увеличивается несущественно по сравнению со временем ожидания редактором действий пользователя. Объем кодов программы не увеличивается при эффективной реализации макроопределений; объем текстов программы может даже уменьшиться, поскольку пробелы в файле (но не на экране) сжимаются.

Автором статьи на механико-математическом факультете МГУ разработан редактор-макрогенератор «МакроМир» для компьютеров серий PDP-11 (в том числе и для УКНЦ) и IBM PC. «МакроМир» для УКНЦ раскрывает макроопределения из пакета «Макро-85». С ним можно также работать на компьютерах серий PDP-11 или IBM PC, используя другой пакет, с последующей сборкой программы на УКНЦ или на PDP-11 (в ОС RT-11 или ее эмуляторе).

Редактирующие возможности «МакроМира» включают возможности широко известного текстового редактора «МикроМир» (также разработанного на мехмате МГУ) и дополнительный сервис по синтаксическому редактированию: вставке, удалению, запоминанию (возможно, с удалением) синтаксических конструкций, откатка этих действий, а также облегченное редактирование колонок операторов и комментариев.

Предлагаемая технология разработки программного обеспечения — использование редактора-макрогенератора совместно с пакетом «Макро-85» — приблизит программирование на ассемблере к процессу мышления разработчика, повысит структурированность и читаемость программ, облегчит их ввод, сократит число ошибок и тем самым ускорит написание и отладку программных систем.

Литература

1. *Сингер М.* Мини-ЭВМ PDP-11: Программирование на языке ассемблера и организация машины. М.: Мир, 1984.

2. *Подольская Н. А.* Редактор-препроцессор для Макро-11. М., 1988. 19 с. Деп. ВИНИТИ 20.01.88, 487—В88.

В. ЖИГУНОВ

«Техноком» представляет

«Электроника МС 0511», или УКНЦ (что расшифровывается как «Учебный компьютер "Научного центра"), получила широкое распространение и выпускается в течение нескольких лет. Она обладает довольно высокими техническими характеристиками по сравнению с другими ЭВМ своего класса, однако эффективность ее использования снижается из-за недостатка программного обеспечения. Базовый комплект поставки не содержит сервисных и прикладных программ, что ставит ее владельцев в крайне затруднительное положение. Использование УКНЦ в школе чаще всего сводится к освоению Бейсика.

Усилия МНПП «Техноком» направлены на исправление этого положения. Здесь разработаны эффективные прикладные и сервисные программы, которые с успехом могут применяться в учебном процессе и других областях. Кроме того, разрабатываются и производятся дополнительные технические средства для УКНЦ — кассеты ПЗУ с прикладными программами, электронный диск, накопитель типа «винчестер» на 20 Мбайт, стенд для ремонта. Они позволяют улучшить технические и эксплуатационные характеристики компьютера.

Графическая обучающая система GUESS — простой и удобный инструмент создания обучающих и контролирующих программ для практически любых учебных дисциплин (математики, химии, иностранных языков и т. д.). Основа системы — редактор, позволяющий нарисовать на экране «картинку» (кадр) в соответствии с выбранным вами сценарием. Урок будет состоять из набора

таких кадров; переходы между ними задаются в зависимости от ответов учащегося. Работая с редактором, можно использовать одновременно и графическую, и текстовую информацию в цвете, рисовать линии, точки, прямоугольники, окружности, проводить масштабирование изображения. Главным отличием GUESS от других подобных программ являются возможности создания движущегося изображения (мультипликации) и работы со звуком (создание любых звуков и нот).

На кадре можно задать несколько областей, которые будут рассматриваться как варианты ответа ученика. В этих областях могут помещаться как текст, так и изображения.

Редактор прост для освоения и не сковывает педагога. Главной задачей последнего становится разработка сценария урока, переложение на ЭВМ трудностей не составляет.

Система имеет средства для быстрой рассылки созданных уроков на места учеников.

Многофункциональный сетевой монитор — незаменимое средство для эффективного использования локальной сети в учебном классе. Он дает возможность работать на РМУ с любыми программами, с которыми можно работать на РМП. Монитор имеет ряд новых возможностей, повышающих эффективность работы преподавателя: автоматическая загрузка на любое РМУ операционной системы по включению питания РМУ, автономная работа каждого РМУ с любыми программами, одновременная (параллельная) загрузка системы на все РМУ (требует около 10 с), параллельная загрузка любой программы на РМУ и на электронный диск этого РМУ, разрешение доступа с РМУ только к назначенным областям дисков (устройств SY:, LD:, MZ:, KD:), что позволяет защитить дискету преподавателя от несанкционированного доступа; наконец, вывод на печать информации с РМУ.

Система проста в эксплуатации. Информация о состоянии каждого РМУ представлена в графическом виде на мониторе преподавателя; отображаются режимы загрузки, записи, чтения. В специальном окне отображается каталог диска, можно выбрать любой файл для загрузки на РМУ. Управление осуществляется с помощью всего нескольких клавиш. Система устойчиво работает при интенсивной эксплуатации учебного класса.

Графические редакторы предназначены для создания и редактирования на экране видеомонитора графической информации. Различаются они степенью насыщенности различными функциями, объемами использования ресурсов ЭВМ, методами формирования данных в памяти и, естественно, способами использования полученного изображения.

Редактор SEG создает монохромные графические объекты размером 32×32 точки, называемые спрайтами. Спрайты можно формировать, редактировать, запоминать в отдельном буфере, а в дальнейшем использовать в прикладных программах. SEG является инструментальным средством для программиста, знакомого с макроассемблером.

Редактор PPG формирует цветное изображение на всем поле экрана, имеет простое управление и может применяться для учебных целей. Предусмотрена возможность сохранения изображений на диске.

Редактор GRUK позволяет наиболее эффективно использовать возможности УКНЦ, прекрасно подходит для создания сложных, насыщенных схем, имеет много функций для формирования и редактирования изображений, ряд качественных отличий от существующих графических редакторов. Вся «картинка» хранится в ОЗУ, что позволяет обрабатывать информацию с большой скоростью; используется полный экран, задействованы служебные строки; хранение изображений в виде команд, а не копии видео-ОЗУ существенно экономит память. Используются большой набор примитивов и сегменты (наборы примитивов), четыре вида линий (сплошная, точечная, пунктирная, штрихпунктирная), «лупа». Текст выводится на экран с установкой размера символа, ориентации и направления строки; при стирании части изображения не остается «дыр». Можно создавать файлы графических библиотек (шрифты, изображения деталей, фрагменты схем).

Использование меню упрощает работу пользователя, на информационное поле постоянно выдается текущее состояние редактора. Вывод на печать возможен в формате А4 или А3.

Универсальный драйвер печати EP. Как известно, в УКНЦ можно загружать «самодельные» шрифты и знаки, подготовленные программами SETGEN, MENFNT или FONTER. Но для их вывода на бумагу необходим драйвер печати, способный использовать нестандартные режимы работы принтера. EP печатает именно те символы, которые мы видим на экране видеомонитора, а не те, которые написаны на клавишах. Он работает с принтерами типа Epson, Robotron, D-100M, имеет удобную систему меню для настройки на различные параметры, может работать и как обычный драйвер.

Электронная таблица UKCALC. Многие пользователи оценили достоинство подобного рода программ для всевозможных вычислений. UKCALC имеет 12 столбцов и 24 строки; в каждую клетку можно записать число, текст (около 30 символов) или формулу (до 30 символов, арифметические операции и некоторые тригонометрические функции). В клетке с формулой выдается результат вычисления.

Можно, например, записать несколько чисел в столбик, а в нижней клетке задать их суммирование. Теперь, изменив какое-либо число, мы мгновенно увидим новую сумму.

Достоинство электронной таблицы — возможность легко проводить любые арифметические вычисления, зависящие от параметров, сразу получая результаты; быстро составить довольно сложный алгоритм вычисления, не используя языки программирования.

Результаты можно хранить на диске или вывести на печать.

Малая издательская система POLYSS. Это, по существу, графический редактор, имеющий возможность обработки текстовой информации. С его помощью можно подготовить страницу текста, используя любые шрифты и даже графику, и, распечатав ее на принтере, получить документ «товарного» вида.

POLYSS включает библиотеку шрифтов и позволяет печатать текст несколькими видами шрифта на одной странице, имеет функции рисования графических примитивов (линий, прямоугольников, закрашенных прямоугольников, окружностей) и закраски областей различными узорами. Она может запомнить, повернуть, сжать или растянуть в любом направлении любую область экрана, позволяет с помощью «лупы» менять прорисовку мелких деталей. Возможна обработка текста, подготовленного в обычном текстовом редакторе.

Благодаря системе меню используется минимум управляющих клавиш, что позволяет быстро освоить POLYSS и обеспечивает простоту работы с ней.

Итог — сочетание удобств текстовых и широких возможностей графических редакторов.

Файловый монитор SAM является функциональным аналогом программы Norton Commander и предназначен для оперативной работы с файлами. Выводит на экран два каталога по 20 файлов, например каталоги верхнего и нижнего дисков; имеет меню основных команд (удаление, копирование, создание, просмотр, трансляция, печать и др.). Для их исполнения необходимо выбрать (с помощью клавиш управления курсором) нужный файл (или группу файлов) и нужную операцию. Таким образом, пользователь легко может управлять работой ЭВМ, даже не зная команд операционной системы.

После выхода из какой-либо программы система автоматически возвращается в SAM и на экране снова видны два каталога.

Процессор настройки SETGEN — универсальный инструмент, способный заменить многие прикладные программы установки параметров УКНЦ. Он позволяет:

программировать шрифты (изменять прорисовку любого символа, создать свою библиотеку и записать ее на диск);

устанавливать режимы накопителей (выбор номера привода, число сторон, время шага головки, время загрузки головки, число секторов);

программировать клавиши-ключи (К1—К5; им присваивается любая последовательность символов или кодов);

устанавливать параметры пароля (его текст и число попыток набора);

устанавливать формат (число символов в строке), стартовые команды (аналогично программам DAY.SAV, STARTS.COM) и др.

SETGEN может создать исполняемый файл, устанавливающий все перечисленные параметры при каждой загрузке ОС. Имеется встроенная подсказка по всем режимам работы программы.

Кассета ПЗУ-Бейсик избавляет от затрат времени на загрузку файла BASIC.SAV на РМУ, позволяет эксплуатировать КУВТ даже при выходе из строя НГМД и сети. Достаточно иметь одну кассету, чтобы загрузить Бейсик на каждую ЭВМ КУВТа. Созданные на занятиях программы можно хранить на магнитофонной кассете.

Кассета ПЗУ-редактор позволяет использовать УКНЦ, даже не имеющую дисковода, как универсальную пишущую машинку. С ее помощью можно подготовить, отредактировать и распечатать любой документ. «Зашитый» в ПЗУ редактор является функциональным аналогом известного редактора NED, но отличается удобной системой меню для выбора режимов работы. Имеется также встроенная справка и программа инициализации электронного диска.

Дополнительный электронный диск — расширение ОЗУ УКНЦ, используемое в качестве имитатора внешней памяти. Он собран на микросхемах КР565РУ7, имеет емкость 1 Мбайт, выполнен в виде отдельного блока и размещается на столе рядом с компьютером. Наличие собственного блока питания позволяет хранить информацию и при выключении питания УКНЦ.

Применение электронного диска позволяет существенно повысить эффективность работы. Улучшается общая надежность, экономится ресурс НГМД, скорость работы с

файлами увеличивается в несколько раз (что чрезвычайно ценно при создании и отладке больших программ).

Диагностический стенд — необходимое условие эффективного и качественного ремонта любой ЭВМ, особенно УКНЦ, имеющей сложную организацию общей шины и включающей заказные большие интегральные схемы.

На основе стенда можно в кратчайшие сроки организовать рабочее место ремонтника и проводить на нем все виды работ по диагностике и регулировке плат, входящих в состав УКНЦ. Стенд умещается в стандартный портфель типа «дипломат», что позволяет проводить ремонт и на территории заказчика. В комплект поставки входят магнитный диск с текстовым обеспечением (включает 44 программы, позволяющих полностью проверить микро-ЭВМ — от простейшего теста на короткие замыкания и обрывы шин до внутренних тестов процессоров, памяти и других узлов) и ремонтная документация: описание микросхем («зашивок») серии КР1801ВП1 и КА1515ХМ1, руководства по среднему ремонту платы вычислителя, сетевого адаптера, контроллера магнитного диска, блока питания.

Для организации рабочего места дополнительно необходимы управляющая ЭВМ «Электроника МС 0511» с монитором и накопителем, осциллограф, вольтметр, блок питания.

Т. КРУГЛИК, В. ТИМАШОВ

Работаем с УКНЦ

В настоящее время идет активный поиск путей и разработка методик применения компьютерной техники в учебном процессе. В связи с этим возникает задача: подготовить учащихся к использованию компьютерной техники в учебной деятельности.

На примере КВУ «Электроника УКНЦ» мы хотим показать, как можно подойти к ее решению, учитывая специфику технических и программных средств.

На основании опыта работы мы сделали вывод, что для применения ЭВМ в учебной деятельности необходимы следующие базовые знания, умения и навыки:

знание структурной схемы ЭВМ,

знание способов представления, обработки и хранения информации в ЭВМ,

умение работать с клавиатурой и другими аппаратными средствами,

знание назначения и структуры операционной системы (ОС),

начальные навыки работы с командным языком операционной системы,

навыки работы с инструментальными программными средствами.

Выбор путей и средств формирования перечисленных знаний, умений и навыков зависит от двух основных факторов:

технических особенностей данного компьютерного класса;

особенностей применяемого программного обеспечения.

По техническим данным микро-ЭВМ «МС 0511», на базе которых создан класс, стоят значительно выше других отечественных микро-ЭВМ учебного назначения. Но, несмотря на это, учебный класс обладает рядом недостатков: низкой скоростью передачи по локальной сети, низкой надежностью дисковода, отсутствием работоспособного сетевого программного обеспечения учебного назначения.

Все это не может не создавать определенных трудностей, которые преодолеваются с помощью специальных программных средств.

Нами разработан пакет «ЛВС УКНЦ», который дополнил и частично заменил программное обеспечение, поставляемое с КУВТом. Он позволил реализовать работу на рабочих местах учеников (РМУ) с редакторами, базами данных, компиляторами, интерпретаторами и другими программами, сохранив совместимость с программными средствами, не использующими графики для компьютеров линии ДВК.

Сетевые возможности пакета поддерживаются следующими компонентами:

программа поддержки локальной сети,

драйверы прямого и последовательного доступа для сети,

МИКРО-ОС,

квази-диски,

командные файлы.

На основе этих программных средств, а также благодаря наличию тренажеров, графического пакета, текстового редактора мы обеспечиваем учащимся возможность получения знаний и умений, соответствующих базовым.

На первых этапах обучения нами ис-

пользуются демонстрационные программы, созданные с помощью специального графического редактора. Особенность редактора состоит в том, что в результате его работы может быть получен исходный текст демонстрационной программы на Паскале.

Навыки работы с клавиатурой закрепляются в результате использования тренажеров различной степени сложности.

Большие трудности возникают при выполнении учащимися практических работ, требующих использования значительных ресурсов ОС. Сюда можно отнести, например, работу с текстовым и графическим редактором, с базами данных и др. Есть два пути для преодоления этих трудностей: первый — создание инструментального класса на базе существующего, т. е. оснащение РМУ контроллерами магнитного диска и дисководами; второй — используя специальные программные средства, загружать на РМУ усеченную версию ОС (МИКРО-ОС).

Первый путь предпочтительнее, но связан со значительными материальными затратами. Второй путь требует наличия «МИКРО-ОС» и методики его использования в учебном процессе.

Это программное средство исключает оверлейный режим в сети и дает возможность организации автономной работы с минимальной версией ОС на каждом РМУ.

На наш взгляд, в минимальный набор команд монитора, достаточный для формирования у учащихся практических навыков работы с файлами, входят следующие команды:

DIRECTORY, COPY, PRINT, DELETE, CREATE, TYPE, R, RUN. Эти навыки, в свою очередь, являются основой для эффективного обучения работе с инструментальными программными средствами.

Заключительным этапом формирования базовых знаний, умений и навыков является изучение редактора текстов. В итоге учащиеся получают возможность практического применения компьютерной техники на примере современных способов

Тема практического занятия	Тексты заданий	Командный файл
1. Каталог файлового устройства, команда DIRECTORY	Прочитать каталог диска, находящегося на устройстве DK; определить максимальный размер свободной области в каталоге диска и указать начальный номер блока этой области; используя команду DIR с ключом OUTPUT, создать файл с именем и типом DIR.HLP на устройстве KSO; подсчитать общее количество блоков, занимаемых файлами типа SYS на устройстве DCO	RMU1.COM
2. Копирование файлов, просмотр текстовых файлов	Используя команды TYPE, COPY, вывести на экран монитора содержимое текстового файла PRIM1.LST, находящегося на KS (выполнить несколько вариантов записи команд, используя умолчания); создать копию предыдущего файла с именем PRIM2 и типом TXT на устройстве DK, проверить наличие вновь созданного файла в каталоге диска; скопировать файл LP.SYS с устройства DCO на устройство KS, изменив имя LP на LP1; скопировать клавиатурный тренажер LITER.SAV с DC1 на SY и запустить его на выполнение.	RMU2.COM
3. Создание файлов с помощью редактора текстов EDIK	С помощью редактора текста, находящегося на устройстве DK создать текстовый файл, содержимым которого является ответ на вопрос, что такое спецификация файла, из чего она состоит; имя файла — первые четыре буквы Вашего имени, тип — TXT; созданный файл скопировать на устройство DC1.	RMU3.COM
4. Работа с файлами	Даны два файла. Отредактировать их, исправив грамматические ошибки и удалив повторяющиеся строки; объединить полученные файлы, используя функции редактора; объединить эти файлы в один, присвоив ему другое имя, с помощью команды монитора.	RMU4.COM

хранения и обработки текстовой информации.

Весь процесс практического обучения мы строим на основе системы заданий, которая, на наш взгляд, обеспечивает учащимся необходимый набор знаний, умений и навыков.

В таблице приведены примеры типовых заданий, предлагаемых учащимся на практических занятиях по темам «Командный язык ОС, команды монитора», «Текстовый редактор EDIK».

Практические работы на основе системы заданий, направленных на формирование базовых знаний, умений и навыков, проводятся нами на занятиях кружков и в УПК. Однако мы считаем, что такие занятия должны найти место в процессе обучения информатике и ВТ. Это позволяет создать у учащихся мотивацию для использования компьютера в учебной деятельности.

Конференция в Троицке

Если в прошлом году вам не повезло и вы не смогли принять участие в конференции «Применение новых компьютерных технологий в образовании», не расстраивайтесь! У вас есть отличный шанс стать ее участником будущим летом. Ожидается участие до 100 иностранных и 300 отечественных специалистов. Эта традиционная международная конференция пройдет в г. Троицке Московской обл. 25—27 июня в Центре информатики «Байтик». Организаторами и спонсорами конференции станут:

Троицкий институт инновационных и термоядерных исследований (TRINITY), калифорнийская организация Computer Using Educators (CUE), Комитет по народному образованию Мособлисполкома, журнал «Информатика и образование».

Предлагаются следующие основные направления обсуждений:

практическое использование компьютерных технологий для обучения (от дошкольного до высшего образования);

компьютерные приложения для специальных дисциплин: экономика и бизнес, экология, настольное издательство, техническое рисование и дизайн и др.;

нетрадиционные виды обучения: телекоммуникационное, дистанционное, видеообучение;

компьютер для учителя и школьной администрации.

Более полную информацию вы можете получить, а также прислать заявки и предложения:

142092 Троицк Московской обл., Сиреневый б-р., 11, «Байтик». Тел. (095) 334-03-67.

ЧТО МОЖЕТ ЭВМ?

Выживет ли RISC?

Еще совсем недавно во всем мире решался вопрос: RISC-архитектура — хорошо это или плохо? С одной стороны, очень ограниченный набор команд, с другой — огромное быстродействие. Вопрос этот уже решен, опытные образцы доказали свою жизнеспособность. Вслед за ними появились мощные рабочие станции и даже RISC-лэптопы.

Однако стандартом де-факто является открытая архитектура персональных компьютеров фирмы IBM. И поэтому полная несовместимость RISC-машин с популярной моделью препятствует их широкому распространению.

Как правило, основные покупатели RISC уже имеют хорошо оборудованные прикладным программным обеспечением персональные компьютеры IBM. Теперь им нужна большая мощность для новых задач, но они не хотели бы терять уже наработанных программ. Одним из путей, позволяющим сохранить привычное окружение, является эмуляция работы Intel-процессора. Задача эта достаточно проста, но мощность RISC-станции настолько велика, что на ней можно имитировать работу сразу нескольких персональных компьютеров в мультизадачном режиме.

Это обстоятельство усугубляется тем, что даже для сравнительно маломощной IBM PC некоторые задачи слишком просты. Работа текстового редактора, например, не требует производительности процессора в 25 миллионов инструкций в секунду (mips).

Второй способ сближения RISC и Intel — обеспечение файловой совместимости. В этом случае RISC-машина считывает файл со стандартной DOS-дискеты, обрабатывает его и вновь возвращает на дискету в стандартном DOS-формате.

И, наконец, третий, наиболее полно обеспечивающий совместимость компьютеров двух видов способ — это использование в рабочей станции RISC платы с процессором Intel 386. «С такой дополнительной платой RISC архитектура не только выживет, но и получит дальнейший толчок в своем развитии»,— считает Brian Rosen, президент фирмы Mars.

Все, что пожелаете

Такое обещание дает покупателям своей продукции японская фирма NEC, и не без оснований. Ее новый лэптоп ProSpeed SX/20 действительно превосходит по своим параметрам многие настольные компьютеры. Но дело здесь не в высочайшей японской технологии, а в оригинальном конструктивном решении, которое позволило разбить компьютер на две части — переносную, обладающую достаточным запасом мощности для работы в дороге, и стационарную, которая доводит производительность и другие характеристики комплекса до профессиональных.

Вот такие матрешки делают в Японии.

Компьютер — видеомагнитофон

Резкая активизация деловой жизни в нашей стране привела к расширению функций средств массовой информации. Телевидение, например, начало использовать такую форму, как телебиржа. Этот очень емкий и оперативный способ передачи информации использует стандартный видеосигнал, позволяющий в каждом телевизионном кадре разместить текстовую информацию об одном объекте купли-продажи. Таким образом, всего за десять секунд эфирного времени перед глазами изумленного телезрителя промелькнет двести пятьдесят полных описаний разнообразных товаров.

Ознакомиться с ними в таком темпе невозможно, даже обладая навыками скорочтения. Но, записав информацию на видеомагнитофон, впоследствии можно внимательно изучить ее в режиме покадрового просмотра.

Новая цифровая видеосистема, созданная совместно двумя японскими фирмами — Intel Japan и Matsushita Graphic Communication Systems,— позволит обойтись без видеомагнитофона. Это устройство, объединяющее в одном корпусе процессор 1750, периферийный процессор и компьютерный контроллер, позволяет запоминать на компактдиске и обрабатывать видео- и аудиоинформацию и воспроизводить ее при необходимости на экране телевизора. Кроме того, с помощью новой системы можно формировать титры и различные видеозаставки для домашнего употребления.

На каждый замок — своя отмычка

Наибольшую надежность защиты программ от несанкционированного копирования обеспечивают электронные замки. Впрочем, их называют обычно ключами, потому что защищенная программа начинает работать тогда, когда этот прибор вставляют в разъем параллельного или последовательного порта компьютера, т. е. «отмыкают замок». Размер прибора — со спичечный коробок; он прилагается к каждому экземпляру программы.

Чтобы взломать такую защиту, нужно не только разобраться в программных ухищрениях, но и разгадать содержание обмена программы с ключом, а для этого нужны как минимум специальные приборы. Впрочем, иногда можно обойтись и без них. Ведь ключи выпускают фирмы, специализирующиеся на защите, а используют (в меру своего разумения) фирмы, специализирующиеся на создании прикладных программ, что иногда приводит к забавным ситуациям. Так, одна из «программистских» фирм реализовала весьма сложный и запутанный протокол проверки наличия ключа, однако результатом проверки было... занесение 0 (если ключа нет или он поддельный) или 1 в некий байт. Таким образом, достаточно, переждав проверку, занести в этот байт 1, чтобы программа заработала.

Бывают ли отмычки проще?

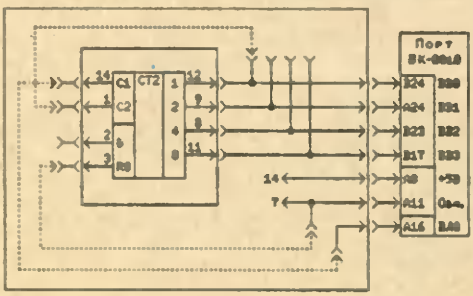
Исследование ИМС с помощью БК-0010.01

На некоторых специальностях Хабаровского техникума железнодорожного транспорта изучается предмет «Основы вычислительной и микропроцессорной техники». Чтобы досконально усвоить логику работы ИМС, необходима отработка принципов их работы на макетах-тренажерах.

Традиционный макет-тренажер стандартен: это блок питания, индикаторы-светодиоды, элементы коммутации, генераторы, осциллограф. Изготовить такой макет трудно и дорого. Мы решили идти нетрадиционным путем.

Сначала разработали программы, позволяющие демонстрировать работу логических элементов И, ИЛИ, НЕ, ИЛИ — НЕ, И — НЕ. При изменении уровней входных сигналов заполняется таблица истинности, одновременно на экране вычерчивается временна́я диаграмма. Уязвимость способа в том, что нет навыка практической работы собственно с ИМС.

Освоив КУВТ-86, мы нашли иное решение этой задачи. Поясним на примере исследования ИМС счетчиков-делителей К155ИЕ2, ИЕ4, ИЕ5. Выполняется макет с коммутационными гнездами и набором шаблонов, на которых изображены условные графические обозначения указанных ИМС (принципиальная схема — рис. 1).



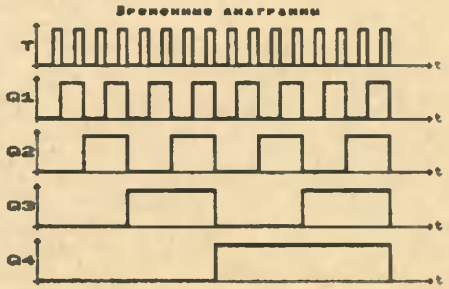
На макет с помощью разъема можно установить любую ИМС. Питание ИМС обеспечивается от БК-0010. Управление работой ИМС осуществляется программно: микро-ЭВМ через порт ввода-вывода подает запускающие импульсы и анализирует состояние триггеров счетчика. Работа с программой сводится к нажатию клавиши **ПРОБЕЛ**, что имитирует подачу входного импульса и приводит к заполнению очередной строки таблицы истинности. Вид таблицы истинности зависит от того, какая коммутация произведена шнуровыми парами на макете (можно получить различные коэффициенты счета ИМС). При желании можно получить твердую копию таблицы. Пример — на рис. 2.

Таблица состояний

Номер импульса	Q1	Q2	Q3	Q4
0	0	0	0	0
1	1	0	0	0
2	0	1	0	0
3	1	1	0	0
4	0	0	1	0
5	1	0	1	0
6	0	1	1	0
7	1	1	1	0
8	0	0	0	1
9	1	0	0	1
10	0	1	0	1
11	1	1	0	1
12	0	0	1	1
13	1	0	1	1
14	0	1	1	1
15	1	1	1	1
16	0	0	0	0

Счет окончен

После заполнения таблицы истинности на экране строится временна́я диаграмма с требуемым коэффициентом пересчета. Пример — на рис. 3.



Перед началом работы с тренажером учащемуся дается задание получить для указанной ИМС временные диаграммы при нескольких коэффициентах деления.

Данная методика исследования ИМС не требует больших материальных затрат на остродефицитную измерительную технику, а по наглядности и простоте обращения с макетом, на наш взгляд, превосходит все известные нам методы.

А. ПОПОВ,
А. КОНДРАТЕНКО

Самомодифицирующиеся программы

1. Полезными являются только четыре режима адресации, использующие программный счетчик (PC) в качестве регистра, а именно: 27 (непосредственный), 37 (абсолютный), 67 (относительный), 77 (косвенно-относительный).

2. После изменения программы требуется ее перекомпиляция.

3. Доступ к ячейкам, содержащим команды, должен быть разрешен только по чтению.

Эти и другие незыблемые правила работы в системе PDP-11, в качестве аналога которой можно рассматривать БК-0010, на самом деле являются достаточно спорными. Их нарушение дает порой новые ценные возможности.

Для начала кратко рассмотрим способы адресации, обратив внимание на некоторые известные вещи.

Регистровый (0) и косвенно-регистровый (1) режимы достаточно тривиальны.

Автоинкрементный (2) и автодекрементный (4) позволяют работать с массивами данных: чисел, кодов символов и т. д. Для этого нужно записать в один из регистров (Rk) адрес первого элемента массива (или следующего за последним) и в цикле обработать весь массив, обращаясь к элементу (Rk)+ или —(Rk).

Косвенно-автоинкрементный (3) и косвенно-автодекрементный (5) режимы позволяют работать с массивами адресов. Обработка массива аналогична предыдущему, а вот что такое массив адресов, разберем подробнее.

Смысл этих адресаций в том, что значение из очередной ячейки массива используется как адрес ячейки, с которой работает команда. В массиве адресов можно записать последовательно адреса регистров внешних устройств (для последовательного обращения к ним), адреса конкретных данных (заранее записанные, случайные или введенные с клавиатуры) и т. д. Не исключается также повторение адресов.

Что касается индексного (6) и косвенно-индексного (7) режимов, они позволяют обращаться к элементу массива данных (или адресов) по его номеру, если адрес начального элемента записать в регистр, а номер (для слов — удвоенный номер) — в ячейку для индекса. Отметим, что индекс (номер) записывается в отдельную ячейку, следующую за командой. К этому очень важному замечанию мы еще вернемся.

Непосредственный (27) режим позволяет записывать значение в ячейку, следующую за командой (например, для начального присваивания). Абсолютный (37) режим задает адрес для обработки, например адрес регистра внешнего устройства.

Адресации относительная (67) и косвенно-относительная (77) позволяют перемещать программу без дополнительной подстройки адресов, так как используют разность адресов (смещение) команды и ячейки, с которой работает команда или в которой находится адрес рабочей ячейки. Здесь также следует отметить, что смещение записывается в отдельной ячейке.

Из оставшихся четырех режимов остановимся подробно на 07 и 17 (47 и 57 действительно бесполезны).

Режим 07 рассмотрим на примере команд MOV и ADD. Если PC используется в качестве источника, то, например, команда MOV PC,DD (DD — приемник) позволит запомнить адрес следующей ячейки в приемнике. Это полезно при работе с подпрограммами, но команда JSR более эффективна. Команда ADD PC,DD и вовсе бесполезна. Если же PC — приемник, то получим следующее. MOV SS,PC (SS — источник) приведет к замене значения счетчика команд на значение источника, что равносильно безусловному переходу по абсолютному адресу. ADD SS,PC увеличит значение PC на значение источника, т. е. источник сыграет роль смещения аналогично командам BR, BEQ и т. д. Но здесь смещение не ограничивается одним байтом,

что делает эту команду полезной в случаях, когда большое смещение не позволяет использовать команды BR, BEQ и др.

Пусть нужно написать команду «если равно 0, то на 530_8 слов». Так как $530_8 > 377_8$, то BEQ использовать нельзя. Придется действовать так:

```
      BNE  A          001002
      ADD  #1260,PC   012707
                      001260
A:    продолжение     ......
```

Замечание: $1260_8 = 2 \times 530_8$. Смещение задается в байтах.

Таким образом, адресация 07 не так уж бесполезна.

Займемся теперь адресацией 17, ради которой и написана статья.

Рассмотрим команду MOV (PC),DD. При ее выполнении процессор:

скопирует в приемник значение из следующей после команды ячейки (в момент выполнения в PC содержится ее адрес);

перейдет к чтению следующей команды;

так как в PC находится адрес ячейки, которая была источником, то ее значение и воспримет процессор как очередную команду.

Вот тут и возникает противоречие: содержимое ячейки должно быть, с одной стороны, значением, а с другой — командой. Трудно не согласиться с Т. С. Фрэнком, когда он в своей книге «PDP-11. Архитектура и программирование», отказав адресации 17 в практичности, переходит к адресации 27, при которой после прочтения значения PC снова увеличивается на 2 и указывает на следующую ячейку, уже с командой.

Зато случай MOV SS,(PC) предоставляет новые возможности, на которые не обратил внимания даже Т. С. Фрэнк. Проследим за выполнением этой команды:

процессор скопирует содержимое источника в следующую после команды ячейку (аналогично предыдущему случаю);

перейдет к чтению следующей команды;

прочтет только что записанное как команду.

Сразу возникают вопросы: зачем записывать команду в ячейку перед ее выполнением, затратив при этом еще команду, если можно было записать ее туда в процессе создания программы? Как быть с той информацией, которая была в ячейке ранее? Где гарантия, что значением источника будет код нужной команды? Что, кроме неприятностей, принесет изменение команд программы в процессе выполнения?

Отвечать будем на все вопросы сразу. Если таким способом записанная команда выполняется один раз, то этот прием бесполезен. Если же выполнение будет многократным (цикл, подпрограмма и др.), то можно ожидать полезных эффектов, если записываться будут разные команды.

Простой пример. Пусть в источнике используется автоинкрементная адресация с регистром Rk; тогда команда MOV (Rk)+,(PC) при каждом проходе цикла запишет в следующую ячейку очередной элемент массива, начальный адрес которого предварительно был записан в Rk. Этот массив логично назвать массивом команд, так как очередной его элемент будет выполняться как команда при очередном проходе цикла. Таким образом, этот прием позволяет использовать циклы с меняющимися командами.

Пример более сложный. В компьютере БК-0010 используются 15 различных режимов работы дисплея. С помощью команды EMT 34 можно определить ССД (слово состояния дисплея), в котором отражены текущие режимы. Поскольку для каждой программы нужны свои режимы, требуется установить их.

Допустим, необходимо установить режимы (если они сброшены) 32 символа в строке (0-й бит), русский регистр (3-й бит), подчеркивание символа (4-й бит), гашение курсора (14-й бит); сбросить режимы (если они установлены) инверсия фона (1-й бит), расширенная память (2-й бит), инверсия символа (5-й бит), индикация СУ (6-й бит), блокировка редактирования (7-й бит), ГРАФ (8-й бит); не изменять остальные режимы (несущественные).

Алгоритм достаточно прост: проверяем наличие (отсутствие) существенного режима и, если нужно сбросить (установить) его, передаем монитору код изменения данного режима с помощью команды EMT 16. Реализовать его можно разными способами:

для каждого существенного режима написать команды проверки и изменения (без цикла);

написать цикл для установки и цикл для сбрасывания режимов;

написать один цикл для всех режимов, используя адресацию 17.

Покажем решение третьим способом.

```
EMT  34          104034 ;SSD в R0
MOV  R0,R1       010001 ;R0 в R1
MOV  #1,R2       012702 ;Маска в R2
                 000001 ;Начальная
     ;                      маска = 1
MOV  #AKOD,R3    012703 ;
                  AKOD  ;Адрес
     ; массива кодов смены режима
MOV  #AKOM,R4    012704 ;
```

```
                        АКОМ  ;Адрес
        ;                       массива команд
    MOV  #17,R5         012705 ;
                        000017 ;К-во
        ;                       повторений цикла
     ; начало цикла             ;
A:  MOVB (R3)+,R0       112300 ;Очередной
        ;   код изменения режима в R0
    MOV  (R4)+,(PC)     012417 ;Очередную
        ;  команду в следующую ячейку
    NOP                 000240 ;..команда..
    BIT  R2,R1          010201 ;Проверка
        ;                       очередного режима
    BEQ  B              001401 ;Проверка
        ;                   на изменение режима
    EMT  16             104016 ;Изменение
        ;                       очередного режима
B:  ASL  R2             006302 ;Очередная
        ;                               маска
    SOB  R5,A           077510 ;
     ; конец цикла
```

```
  Массив кодов
  Байты      Слова
235 233    116633
016 214    007214
234 237    116237
204 202    102202
226 225    113225
000 227    000227
000 000    000000
000 232    000232
```

```
   Массив команд
XOR  R2,R1    074201
NOP           000240
NOP           000240
XOR  R2,R1    074201
XOR  R2,R1    074201
NOP           000240
NOP           000240
NOP           000240
NOP           000240
BR   B        000403
BR   B        000403
BR   B        000403
BR   B        000403
BR   B        000403
XOR  R2,R1    074201
```

Рассмотрим работу программы с массивом команд. В нем только три различные команды; их порядок зависит лишь от того, какие режимы нам нужны, и может быть произвольным. Команда NOP соответствует случаю, когда режим не нужен: последующие команды сбросят данный режим, если он установлен (EMT 16), и не будут изменять, если сброшен (BEQ B). Команда XOR R2,R1 (случай нужного режима) изменяет в R1 значение очередного бита на противоположное (R2 — текущая маска), в результате чего последующие команды будут действовать наоборот: установят данный режим, если сброшен (EMT 16), иначе оставят (BEQ B). В случае несущественных режимов используется команда BR B, которая позволяет сразу перейти к следующей маске и следующему режиму.

Мы получили программу, постоянно изменяющуюся в процессе работы. При этом экономится память и немного замедляется выполнение (так же, как и при замене циклом одинаковых команд).

Такая подстановка возможна только для одновременных команд. В других случаях можно применить адресацию 67, позволяющую менять содержимое ячейки заранее, а не непосредственно перед выполнением. Можно также менять индекс в индексной адресации (он находится в отдельной ячейке).

Пример 1.

```
    MOV  #ADR,R1     012701
                      ADR   ;Начальный
        ;                 адрес массива
A:  ......................
                            ;Расчет
        ;                       индекса
    MOV  SS,B        01SS67 ;Запись
        ;                       индекса
                     ...... ;Смещение
    ADD  @(R1),DD    0667DD ;Работа
        ;       с конкретным элементом
        ;                       массива
B:                   000000 ;Индекс
        ;         (меняется при записи)
    ......................
    BEQ  A           ...... ;Варианты:
        ;                 BNE, SOB, ...
```

Пример 2. Пусть при каждом проходе цикла подставляются команды разной длины:

```
MOV  A,B             016767
                     ..... ;Смещение
     ;                       к A
                     ..... ;Смещение
     ;                       к B
MOV  @#177562,R4     013704 ;Чтение
                     177562 ;кода из
     ; регистра данных клавиатуры
ADD  R5,R2           060502
NOP                  000240
```

Решение.

```
    MOV  #ADR,R1     012701
                      ADR   ;Начальный
        ;          адрес массива команд
A:  ......................;Начало
        ;                           цикла
    MOV  (R1)+,B     012167
                     ..... ;Смещение
        ;                       к B
    MOV  (R1)+,C     012167
                     ..... ;Смещение
        ;                       к C
    MOV  (R1)+,D     012167
                     ..... ;Смещение
        ;                       к D
B:                   000000 ;Три ячейки
C:                   000000 ;для под-
D:                   000000 ;становки
        ;                       команды
    ......................
    BEQ  A                   ;Варианты:
        ;                 BNE, SOB, ...
```

```
Массив команд (цифровые метки указывают
 номера команд и в программе не нужны)
1:   MOV  A,B          016767
                       ..... ;Смещение
          }                        к A
                       ..... ;Смещение
          }                        к B
2:   MOV  @#177562,R4  013704 ;Чтение
                       177562 ;кода из
          }  регистра данных клавиатуры
     NOP               000240
3:   ADD  R5,R2        060502
     NOP               000240 ;Вариант:
          }                  - BR+1 000401
     NOP               000240
4:   NOP               000240 ;Вариант:
          }                    BR+2 000402
     NOP               000240
     NOP               000240
5:   .....................;Далее
          }                    аналогично
```

Этим, конечно, не исчерпываются возможности применения необычных адресаций и программ-хамелеонов, но в небольшой статье описать все случаи невозможно.

Ю. ГИМАТОВ

Обработка прерываний по клавише СТОП средствами Бейсика

Изменение действия клавиши **СТОП** — прием, широко используемый при программировании в машинных кодах. Реализация его проста: адрес обработки прерывания по клавише **СТОП** (он хранится в ячейке 4) заменяется на какой-либо другой (например, начала программы пользователя или ее фрагмента).

Как оказалось, Бейсику тоже вполне по силам «приручить» клавишу **СТОП**. Теоретические предпосылки взяты из статьи А. и В. Авсеевых «Особенности транслятора с языка Бейсик для БК-0010-01» (ИНФО. 1990. № 2.), которым автор выражает огромную благодарность за их замечательную работу, и сводятся к следующему.

Исполняемая Бейсик-программа хранится в памяти БК в виде «шитого кода» — последовательности адресов подпрограмм ПЗУ, реализующих команды Бейсика, и необходимых для работы этих подпрограмм данных. Начальный адрес расположения «шитого кода» задается четным числом, равным или превышающим на 1 хранящееся в ячейке 1026* (т. е. его можно определить как PEEK(1026%)+PEEK(1026%) MOD2%). Управление от одной подпрограммы к другой передается через регистр R4: в конце каждой подпрограммы ПЗУ стоит команда JMP @(R4)+, т. е. в R4 находится адрес адреса следующей подпрограммы. Изменив содержимое R4 в процессе выполнения Бейсик-программы, мы тем самым изменим порядок ее выполнения.

Пусть, например, надо, чтобы при нажатии на **СТОП** Бейсик-программа начинала работу с самого начала. Из сказанного ясно, что для этого следствием нажатия на клавишу **СТОП** должны быть засылка в R4 адреса начала «шитого кода» программы и выполнение команды JMP @(R4)+, что обеспечивается добавлением в начало программы четырех строк:

```
10 POKE 512%,5572% 'код
                    команды MOV (PC)+,R4
20 POKE 514%,PEEK(1026%)+
 PEEK(1026%)MOD2% 'адрес адреса начала
30 POKE 516%,92% 'код
                     команды JMP @(R4)+
40 POKE 4%,512% 'новый адрес
                    обработки прерывания
```

А если при нажатии на клавишу **СТОП** должен произойти переход не к началу программы, а к какому-то ее фрагменту?

Оставим в начале программы те же четыре строки, слегка изменив строку 20:

```
20 POKE 514%,1000%
```

Число 1000 в ней ничего не значит — оно просто резервирует место для адреса начала фрагмента, который мы еще не знаем. Чтобы узнать его, пометим начало нашего фрагмента каким-либо характерным числом, которое мы сможем легко найти, анализируя содержимое ОЗУ; адрес, следующий за адресом этого числа, и будет искомым.

В качестве маркеров удобно использовать операторы END, STOP, BEEP, CLS, CLOSE («шитый код» соответственно —9774, —9748, —8718, —8706, —6738).

Расположим после нашей программы строки, определяющие адрес маркера:

* Здесь и далее все числа даны в десятичной системе счисления.

```
10000 FOR J%=1500%TO16382%ST2%
 'просмотр памяти от 1500 до конца ОЗУ
10010 IF PEEK(J%)=-9774%TH?J%
                    'маркером выбран END
10020 NEXT J%
```

Выполним их.
RUN 10000

В результате может получиться довольно внушительная цепочка адресов, среди которых выделить нужный нелегко. Поэтому пометим начало нашего фрагмента не одним, а тремя одинаковыми маркерами (в данном случае — тремя операторами END, идущими подряд). Теперь, увидев в цепочке адресов три последовательных числа, можно смело сказать, что нужный адрес найден.

Но это еще не все. Нехорошо оставлять программу, «засоренную» вспомогательными элементами, поэтому удалим все три маркера и строки 10 000—10 020, предварительно запомнив адрес первого из маркеров (меньшее из трех последовательных чисел) и посмотрев содержимое ячейки 1026.

После удаления «мусора» произведем «утряску» программы командой REN и вновь посмотрим содержимое ячейки 1026. Там будет число меньше предыдущего. Их разность даст «поправку на удаление»; вычтя ее из первого маркера, получим нужный адрес. Осталось занести его в строку 20 вместо числа 1000, вызвав ее на редактирование командой .20.

При изменении длины программы происходит смещение «шитого кода» в памяти, поэтому перечисленные действия надо производить с уже отлаженной программой.

Рассмотрим пример. Текст программы:

```
10 POKE 512%,5572%
20 POKE 514%,1000%
30 POKE 516%,92%
40 POKE 4%,512%
50 CLS
60 ?"A";
70 J%=J%+1%
80 GOTO 60
81 END
82 END
83 END
90 CLS
100 ?"Выдано"J%"'A'"
110 POKE 4%,-24420%
120 END
10000 FOR J%=1500%TO16382%ST2%
10010 IF PEEK(J%)=-9774%TH?J%
10020 NEXT J%
```

Найдем адрес фрагмента, записанного в строках 90—120:

```
RUN 10000
1798
1800
1802
1842
1900
```

Нужное нам число — 1798.

```
?PEEK(1026)
 1730
DEL 81-83
DEL 10000-
 REN
?PEEK(1026)
 1676
?1798-(1730-1676)
 1744
```

Нашли адрес, помещаем его в строку 20:

```
.20
20 POKE 514%,1000%
20 POKE 514%,1744%
```

Окончательный вид программы:

```
10 POKE 512%,5572%
20 POKE 514%,1744%
30 POKE 516%,92%
40 POKE 4%,512%
50 CLS
60 ?"A";
70 J%=J%+1%
80 GOTO 60
90 CLS
100 ?"Выдано"J%"'A'"
110 POKE 4%,-24420%
120 END
```

Д. ЯКОШВИЛИ
Москва

Псевдопараллельное исполнение программ на БК-0010

Используя возможность прерывания после каждой команды, если установлен бит T слова состояния процессора, можно исполнять несколько кодовых программ «параллельно», когда после исполнения нескольких команд одной программы исполняются несколько команд второй, затем третьей и т. д. Так как время исполнения небольшого числа команд процессора достаточно мало, то подобный прием дает эффект почти такой же, как если бы использовалось несколько менее мощных, чем реальный, процессоров.

Расскажем об этом методе подробнее.

Слово состояния процессора (PSW) БК-0010 имеет адрес 177776. Его младший байт выглядит так:

177776: | P | P | P | T | N | Z | V | C |

Для нас сейчас важен бит T. Обычно он равен 0, если же в нем записана единица, то после исполнения каждой команды происходит прерывание по вектору 14, т. е. выполняются следующие команды:

```
MOV @#177776,-(SP)
MOV PC,-(SP)
MOV @#14,PC
MOV @#16,@#177776
```

После этого можно совершить какие-то действия и возвратить управление основной программе командой RTT, которая выполняет следующие действия:

```
MOV (SP)+,PC
MOV (SP)+,@#177776
```

Исполнится следующая команда основной программы, после чего снова возникнет прерывание по вектору 14 и т. д.

Установку бита T в 1 обычно используют для пошагового исполнения кодовых программ при отладке. Мы же используем этот прием для организации «параллельного» исполнения программ, например, так.

Поместим в ОЗУ N программ и N специальных мониторов, работающих следующим образом. В момент работы первой программы бит T=1, @#14=⟨адрес начала первого монитора⟩. После исполнения одной команды первой программы произойдет прерывание и управление перейдет к первому монитору, который сохранит значения регистров R0—R5, PC и PSW, а затем активизирует данные, необходимые для исполнения второй программы: зашлет в регистры процессора значения, нужные второй программе, в стек — соответствующие ей значения PC и PSW, а в @#14 — адрес начала второго монитора. После этого выполняется команда RTT, возвращающая управление, но уже не первой, а второй программе, поскольку значение PC в стеке было подменено монитором.

Второй монитор аналогично сохраняет регистры второй программы, а восстанавливает регистры третьей и т. д. вплоть до N-го монитора, который сохраняет регистры своей программы, а восстанавливает регистры первой и засылает в@#14 адрес первого монитора.

В результате в N основных программах выполнится по одной команде, причем каждая выполнится так, как если бы других программ не было (не считая уменьшения быстродействия и ресурсов памяти).

Но быстродействие нужно все же посчитать. Сравним «сольное» исполнение программы и «параллельное» вместе с другими N — 1 программам. Пусть время выполнения любой команды равно t. Тогда в первом случае на одну команду тратится время t, а во втором — не менее $tN+ktN$, где k — число команд в мониторе. Следовательно, происходит замедление по крайней мере в $N(k+1)$ раз.

В испытываемых мониторах было 18 команд, замедление при работе двух программ было в 38 раз, не считая времени на операции по прерыванию.

Чтобы ослабить этот эффект, можно выполнять из каждой программы не одну команду, а серию из S команд, в течение которой дополнительное время уходит лишь на приращение счетчика серии, проверку его и команду PTT (обозначим число этих дополнительных команд $k1$). Теперь время исполнения S команд каждой из N программ равно $N(S+k1)t+N(k+1)t$ (к k добавляется единица, потому что при передаче управления другому монитору надо снова выставить счетчик).

Итак, при исполнении S команд каждой из N программ по одной тратится время, равное $tSN(k+1)$, а при исполнении сериями по S команд — $tN(S+k1+k+1)$. В первом случае общее время превосходит полезное (tSN) в $k+1$ раз, во втором — в $(S+k1+k+1)/S$ раз.

При $k=18$, $k1=3$, $S=22$ в первом случае превышение реального времени над полезным в 19 раз, во втором — в 2 раза.

Рассмотрим реализацию этого алгоритма на БК-0010.

```
;10-Jan-91
; Монитор для параллельного исполнения
; двух программ

START: MOV @VEC,R1        ;Пусковая инициализация
       MOV @5,R0
1:     MOV @20,@(R1)+
       SOB R0,1
       MOV @100274,@@4    ;По стопу
       CLR @@6
       MOV #INT1,@@14     ;По T-разряду
       CLR @@16
       MOV @20,-(SP)      ;ССП для 1-й программы
       MOV @20,PSR2       ;ССП для 2-й программы
       MOV @10000,-(SP)   ;Стартовый адрес
                          ;      для 1-й программы
       MOV @17000,PC2     ;Стартовый адрес
                          ;      для 2-й программы
       RTT

VEC:   .@12.@22.@26.@32.@36

INT1:  DEC COUNT          ;Прерывание 1-й
       BNE 1              ;              программы
       MOV @30,COUNT
       MOV (SP)+,PC1
       MOV (SP)+,PSR1
       MOV R0,R01
       MOV R1,R11
       MOV R2,R21
       MOV R3,R31
       MOV R4,R41
       MOV R5,R51
       MOV R02,R0
       MOV R12,R1
       MOV R22,R2
       MOV R32,R3
       MOV R42,R4
       MOV R52,R5
       MOV PSR2,-(SP)
       MOV PC2,-(SP)
       MOV #INT2,@@14
1:     RTT

INT2:  DEC COUNT          ;Прерывание 2-й
       BNE 1              ;              программы
       MOV @30,COUNT
       MOV (SP)+,PC2
       MOV (SP)+,PSR2
       MOV R0,R02
       MOV R1,R12
       MOV R2,R22
       MOV R3,R32
       MOV R4,R42
       MOV R5,R52
       MOV R01,R0
       MOV R11,R1
       MOV R21,R2
       MOV R31,R3
       MOV R41,R4
       MOV R51,R5
       MOV PSR1,-(SP)
       MOV PC1,-(SP)
       MOV #INT1,@@14
1:     RTT

PC1:   .@@                ;Состояние 1-й
PSR1:  .@@                ;              программы
R01:   .@@
R11:   .@@
R21:   .@@
R31:   .@@
R41:   .@@
R51:   .@@
PC2:   .@@                ;Состояние 2-й
PSR2:  .@@                ;              программы
R02:   .@@
R12:   .@@
R22:   .@@
R32:   .@@
R42:   .@@
R52:   .@@

COUNT: .@30               ;Размер серии команд

END
```

Рассматривался случай работы 2 программ, соответственно вспомогательная программа состояла из 3 частей: стартовой части и двух мониторов.

Стартовая часть (метка START) предназначена для установки векторов прерываний (вернее, их вторых слов) по Т-разряду, СТОПу, резервному коду, команде IOT, аварии питания, командам EMT и TRAP. Кроме того, устанавливается холодный запуск монитора по нажатию СТОП и заносится адрес первого монитора (INT1) в @#14, а также заполняются начальные PC и PSW двух программ. Затем управление передается первой команде с одновременной установкой Т-бита с помощью команды RTT.

Программы исполняются сериями по 30 команд. Это реализуется с помощью ячейки COUNT. Мониторы INT1 и INT2 используют ячейки PC1—2, PSR1—2, R01—R51, R02—R52 для хранения состояний программ 1 и 2. Программа написана на ассемблере МИКРО1103Б и не претендует на оптимальность — главным было продемонстрировать идеи метода.

Толчком к применению изложенного подхода было желание озвучивать программы, т. е. добиться параллельно исполнения программы, например, игры и программы, исполняющей мелодию. В экспериментах в качестве основной бралась программка, опрашивающая регистр 177716 и при выявлении нажатии клавиши печатающая код из @#177662. При S=64 и более (до 512) удавалось получить вполне приемлемое звучание, правда при уменьшении количества итераций в циклах типа SOB (они применяются в подпрограммах, генерируемых МЕЛОМАНом) в 2—10 раз.

При большом размере серий начинает чувствоваться прерывистость в работе. Кроме того, поскольку время исполнения разных команд разное, это сказывается на звучании.

Другие возможные области применения: синтез на БК многоголосой музыки, т. е. одновременное исполнение нескольких музыкальных программ;

в играх одна из программ может заниматься обслуживанием героя игры, другая — создавать игровой фон, обслуживая другие объекты;

если в игре участвуют двое, то одна программа может обслуживать одного игрока, вторая — другого, а третья — создавать фон;

в системных программах одна из программ может обслуживать внутренний таймер.

В. БУЛИТКО

Подключение блока клавиатуры МС 7008.01 к БК

На прилавках магазинов «Электроника» недавно появились блоки клавиатуры «Электроника МС 7008.01», практически лишенные тех недостатков, которые столько лет досаждали пользователям БК. Новая клавиатура имеет очень легкий ход, избавлена от дребезга клавиш и с минимальными усилиями может быть встроена в уже имеющийся корпус ПЭВМ.

Технические характеристики клавиатуры:

мощность коммутируемого сигнала 10^{-2} ВА;

количество клавиш — 74;

наработка на отказ — 2×10^6 циклов;

усилие нажатия — 1,2 Н;

ход клавиши 3,5—4,5 мм;

частота нажатий на клавишу — 10 Гц.

Проще всего подключить новый блок клавиатуры к плате вычислителя БК, отрезав от старой клавиатуры два шлейфа проводников с вилками XT1 и XT2 и подпаяв их к разъемам блока в соответствии с приводимой таблицей.

Оставшийся на старой клавиатуре пьезодинамик следует осторожно выпаять и подсоединить к контактам 7 и 5 разъема XT2 МС 7008.01.

Разъем XT1 МС 7008.01	Разъемы шлейфов
1	XT2:15
2	XT1:1
3	XT1:2
4	XT1:3
5	XT1:5
6	XT1:6
7	XT1:11
8	XT1:7
9	XT1:9
10	XT1:10
11	XT1:4
12	XT1:8

Разъем XT2 МС 7008.01	Разъемы шлейфов
1	-
2	XT2:9
3	XT2:10
4	XT2:14
5	XT2:8
6	XT2:4
7	XT2:5
8	XT2:2
9	XT2:3
10	XT2:7
11	XT2:6
12	-

Разъем XT3 МС 7008.01	Разъемы шлейфов
1	-
2	XT1:15
3	XT2:11
4	XT2:1
5	XT1:14
6	XT1:13

Компактность и конструктивная простота МС 7008.01 позволяет использовать его в качестве дополнительной (выносной) клавиатуры; правда, при этом приходится решать две новые проблемы — где достать достаточно гибкий и легкий 27-жильный соединительный шлейф и как сделать корпус, но зато сколько удобств появляется потом!

И. ПАНЧЕНКОВ

Параллельные процессы на БК-0010

Нередко бывает желательно, чтобы во время выполнения программы играла музыка или, к примеру, мигал курсор. Этого можно добиться, вставив после каждых 100 команд команду TRAP с аргументом, соответствующим высоте звука для случая с мелодией, и написав соответствующую программу обработки прерывания по вектору 34.

Однако так невозможно добиться, например, мигания курсора во время выполнения системных EMT-подпрограмм. Преодолеть это ограничение можно, используя прерывание по вектору 14 и внутренний таймер БК, подробно описанный в «ИНФО» 1—90.

Вот пример подпрограммы, заставляющей мигать курсор, в том числе и во время выполнения EMT 6 (во время остальных EMT мигание прекращается).

```
;Текст программы пригоден для систем МИКРО18 и
; старше. В случае использования более ранних
; версий вводите только три первые буквы имен
; меток и констант
```

```
PERIOD=177706
DATA=177710
UPR=177712
PUSK=160
;*** Инициализатор ***
START: MOV #1000,@#PERIOD ;Занесение периода в
                          ;          регистр
       MOV #PUSK,@#UPR    ;Запуск таймера
       MOV #BEGIN,@#14    ;Занесение адреса
                          ;       п/п обработки
       BIS #20,@#6        ;Установить Т-разряд
       BIS #20,@#12       ; для всех прерываний
       BIS #20,@#36
       MOV @#30,SAVEMT    ;Сохранить старый
                          ;          вектор EMT
       MOV #EMT,@#30      ;Активизировать
                          ;       EMT-диспетчер
       BPT                ;Выполнить п/п
                          ;          обработки
       JMP @#100400       ;Выход в пусковой
                          ;             монитор
;*** П/п обработки прерывания ***
BEGIN: CMP @#DATA,#600    ;Прошло ли время?
       BPL RET            ;Нет - продолжить
                          ;             работу
       MOV @#32,-(SP)     ;Сохранить ССП
       BIC #20,@#32       ;Очистка Т-бита
       MOV R0,-(SP)       ;Сохранить R0
       MOV #232,R0        ;Инвертировать курсор
       EMT 16
       MOV (SP)+,R0       ;Восстановить R0
       MOV (SP)+,@#32     ;Восстановить ССП
       MOV #PUSK,@#UPR    ;Сброс и запуск
                          ;             таймера
RET:   BIS #20,2(SP)      ;Установить Т-бит
       RTT                ;Возврат из
                          ;          прерывания
;*** Диспетчер EMT ***
EMT:   MOV R5,-(SP)       ;Сохранить R5
       MOV 2(SP),R5       ;(Адрес EMT)+2 - в R5
       MOV -(R5),CMD      ;EMT - в ячейку CMD
       MOV (SP)+,R5       ;Восстановить R5
       MOV SAVEMT,@#30    ;Восстановить
                          ;          вектор 30
       CMP CMD,#104006    ;Это EMT 6?
       BNE CMD            ;Нет - выполнить
       BIS #20,@#32       ;Разрешить прерывание
                          ;
                          ;
                          ;
                          ;          по Т-биту
CMD:   NOP                ;Выполнить EMT
       MOV @#EMT,@#30     ;Адрес диспетчера
                          ;             - в 30
       BIC #20,@#32       ;Запретить прерывание
                          ;          по Т-биту
       RTT                ;Возврат

SAVEMT: .=0               ;Область для
                          ;сохранения вектора 30
END
```

Программа состоит из трех частей: инициализатора, диспетчера EMT и собственно подпрограммы обработки прерывания. Инициализатор выполняет начальный запуск таймера, сохранение вектора прерывания по EMT, занесение адреса диспетчера EMT в соответствующий вектор и установку Т-битов в ССП всех векторов прерываний. Диспетчер EMT разрешает отладочное прерывание только на время работы EMT 6 (это необходимо для нормальной работы с магнитофоном). Подпрограмма обработки прерывания по вектору 14 определяет, прошло ли необходимое время, и если да, то выдает команду на инвертирование курсора.

В случае использования данной подпрограммы в программе, не использующей запросы EMT или самостоятельно запрещающей прерывание по вектору 14 на время их выполнения, диспетчер EMT и соответствующие команды в инициализаторе можно удалить.

К сожалению, на некоторых БК внутренний таймер неисправен, поэтому приходится прибегать к различным ухищрениям. К примеру, можно подсчитывать число выполненных машинных команд и при достижении им определенного значения выполнять необходимые действия.

На основе системного таймера и прерывания по вектору 14 автором разработана программа МУЗОН, после компоновки с музыкальной подпрограммой редактора МЕЛОМАН 2 образующая перемещаемый блок исполнения мелодии параллельно выполнению основной программы. Существует версия МУЗОНа и с подсчетом числа выполненных команд.

Теоретически возможно распараллеливание до 5 процессов, но реально уже при 3 процессах работа значительно замедляется.

Автор благодарен Шишкину, автору игры KING'S VALLEY, а также авторам игры DIGGER за идею параллельного выполнения различных процессов.

Все заинтересовавшиеся данной темой, а также желающие получить МУЗОН могут написать по адресу: 241047, Брянск-47, а/я 109, брянский клуб пользователей БК.

Р. АСКЕРОВ

ЦЕНТР
СОЦИАЛЬНО-ПЕДАГОГИЧЕСКИХ ИННОВАЦИЙ

предлагает вниманию методистов и учителей информатики

новый тип педагогических программных средств

Интегрированная предметная среда

В качестве первого образца этого типа учебных программных средств мы готовы представить вам *предметно-ориентированную среду по информатике* для 6—7 классов.

Использование ее в учебном процессе позволит решить актуальную для компьютеризации общего среднего образования задачу развития индивидуальных способностей учеников и подготовки их к жизни в информационном обществе.

С идеологией курса информатики можно более подробно ознакомиться, прочитав статью «Методические проблемы преподавания информатики в 5—7 классах» в «ИНФО» № 6, 1989 г.

В состав комплекса входят:

учебник, методичка — 1 для учителя,
компьютерная тетрадь для ученика,
программная среда, методичка — 2.

При оптовой закупке комплекса на уровне области, города, района возможна организация (бесплатно) курсов по подготовке учителей с выездом методистов-разработчиков.

Среда реализована на наиболее современных типах используемой в школах техники: «Ямаха», «Корвет», УКНЦ, IBM PC.

В стадии завершения находится разработка *сред* с другой предметной ориентацией:

«математической» (для 6 и 7 классов),
«физической» (для 9 и 10 классов),
«химической» (для 9 и 10 классов),
«языковой» (для 6 класса).

Все среды имеют *унифицированный интерфейс*, что облегчает их освоение как учащимися, так и учителем.

Но максимально эффективное использование этих новых средств невозможно без опоры на более раннее, чем сегодня, формирование у учащихся навыков работы на компьютере.

Пакет «Администратор»

Представляет собой *систему* из трех модулей:

составление расписания,
база данных «Школа»,
внутришкольный контроль.

Пакет рассчитан на применение в средних школах, ПТУ, техникумах и снабжен подробным описанием технологии работы с ним (включая примеры) и шаблоны для облегчения начального заполнения системы.

Более подробную информацию по интересующим вас программным средствам (состав и функциональное назначение программ, входящих в среду, характеристики пакета **«Администратор»**, сроки готовности новых разработок) можно узнать по адресу:

129110, Москва, пр. Маркса, д. 18, ауд. 217,
лаборатория «Новых технологий обучения».
Контактный телефон для справок: 403-32-96.

А. ФЕДОТОВ

Применение электронных таблиц

В различных учебниках [1—4] даны примеры применения электронных таблиц, каждый из которых не раскрывает всех возможностей темы. Поэтому, мне кажется, необходимо объединить их.

Первый пример можно взять из учебника [2]: заполнение сметы затрат на школьные принадлежности:

A	B	C	D
Наименование	Цена	Количество	Сумма
1. Тетради	10	10	100
2. Карандаши	7	5	35
3. Ручки	70	4	280
4. Резинки	10	2	20
5. Итого		21	435

или из [3]: составление сметы туристического похода, смета инвентаризации дисплейного класса, а также определение оптимальной входной платы в видеокафе из [1].

Для самостоятельной работы учащимся, как показал опыт, лучше предложить более реальную задачу. Например, составление сметы месячных расходов на питание, как на одного человека, так и на семью.

A	B	C	D	E
Наименование	Цена	Количество	Сумма на одного	Сумма на семью

После набора значений и подсчета сумм в столбцах *D* и *E*, а также итоговых сумм дается задание скорректировать полученные суммы в большую или меньшую сторону, а также увеличить или уменьшить количество наименований продуктов. Список продуктов лучше всего подготовить каждому ученику дома.

Третий пример можно взять из [1]: «Прогнозирование численности населения в 2000 г.». Модель численности описывается функцией

$$F(t)=a*EXP(b*t),$$

где коэффициенты a и b для каждого государства свои и нам не известны. Понятно, что если $a>1$, то численность населения растет, если $a<1$, то уменьшается. В учебнике приводится пример заполнения таблицы, если в ней есть функция экспоненты. Рассмотрим случай, когда такой функции нет (как в электронной таблице для БК-0010 *CALC*2,3, автор — А. В. Бакерин, Москва). Для заполнения таблицы воспользуемся тем, что:

$$\begin{cases} F'(t)=b*F(t) \\ F(0)=a \end{cases}$$

Тогда можно применить метод ломаных Эйлера и получить:

$$\begin{cases} Y(k+1)=b*Y(k) \\ Y(0)=a \end{cases}$$

где $Y(k)$ — численность населения в k-том году. Тогда таблицу можно заполнить так:

A	B	C	D
1 *a*	реальные	*A*1	*ABS* (*B*1—*C*1)
2 *b*	числа	*C*1**A*2	*ABS* (*B*2—*C*2)
3		*C*2**A*2	*ABS* (*B*3—*C*3)
4		*C*3**A*2	*ABS* (*B*4—*C*4)
»	»	»	»
»	»	»	»
»	»	»	»

где в ячейке *A*1 — коэффициент a, в ячейке *B*1 — коэффициент b. Столбец *B* заполняется числами из справочника. В столбце *C* получится расчетная численность, а в столбце *D* — отклонения расчетной от реальной по модулю (погрешность). Для чисел, приведенных в [1], получится следующая таблица:

	A	B	C	D
1	264.4	264.4	*A1*	*AB•(B1—B1)*
2	*b*	266.6	*C1•A2*	*AB•(B2—C2)*
3		268.8	*C2•A2*	*AB•(B3—C3)*
4		272.1	*C3•A2*	*AB•(B4—C4)*
5		273.8	»	»
6		276.5	»	»
7		278.8	»	»
8		281.5		
9				*MAX(D1...D8)*

В колонке *B* — реальные числа за 1980—1987 гг. Остается только подобрать такое значение ячейки *A2*, чтобы число в ячейке *D* получилось как можно меньше. После этого необходимо продолжить колонку *C*, и мы получим прогноз на любой год.

Электронная таблица может использоваться также не только для прогнозирования, но и для моделирования и расчета многих временных процессов: равноускоренное движение (физика), размножение (биология), скорость реакции (химия). Также ее можно использовать для решения однотипных задач с различными исходными данными и для ведения школьного бухгалтерского учета.

Литература

1. *Гейн А. Г. и др.* Основы информатики и вычислительной техники. Свердловск: Изд-во Уральского университета, 1989.

2. *Каймин В. А. и др.* Основы информатики и вычислительной техники: Пробное учебное пособие. М.: Просвещение, 1989.

3. *Сапрыкина Г. А. и др.* Основы информатики и вычислительной техники для работников народного образования: Методические рекомендации для преподавателей. Новосибирск, 1987.

4. *Тихонов А. Н., Костомаров Д. П.* Рассказы о прикладной математике. М.: Наука, 1979.

Н. КАЗУЛИН, А. ХРОМОВ

Преподавание предмета «Основы технического творчества»

Пристальное изучение организации учебного процесса в средних специальных учебных заведениях технической ориентации позволило выявить методологический дефект подготовки специалистов, заключающийся в следующем: начиная со II курса учащиеся изучают черчение, материаловедение, микроэлектронику и вычислительную технику и еще целый ряд дисциплин, казалось бы, в достаточной мере наполняющих знаниями создателя новой техники. Но все эти дисциплины не объединены творческим началом — методологией специальности, что ведет к крайне низкому использованию знаний общетехнических дисциплин, физических и химических эффектов, и в итоге учащихся учат технологии копирования. Учащийся копирует, решая задания по различным предметам, копирует при курсовом проектировании, копирует, выполняя дипломное задание.

Поскольку техническое творчество всегда являлось важнейшим средством повышения качества подготовки молодого специалиста и воспитания творческой личности, то для устранения этого дефекта в нашем техникуме с 1989 г. был введен факультативный курс предмета «Основы технического творчества». В то время такой предмет в техникумах не преподавался; литературы было мало, поэтому пришлось самостоятельно составлять программу предмета (некоторая ее часть была взята из книги А. И. Фурсенко и др. «Основы научно-технического творчества»), определять его цели и задачи.

В соответствии с разработанной программой весь курс составляет 72—78 ч, учащиеся знакомятся с основными способами фиксации творческих решений (чертеж, схема, технический рисунок, ведение текстовых документов), основами патентоведения и информационной деятельности, правилами составления заявки на изобретения и рацпредложения. Большое место в программе отводится изучению методов активизации творческих процессов. Учащиеся знакомятся с ассоциативными методами, методом морфологического анализа, мозгового штурма, алгоритмом решения изобретательских задач. В программе предусмотре-

но выполнение практических работ.

В 1990 г. было решено изменить структуру предмета, уделив большое внимание изучению ТРИЗ (теории решения изобретательских задач). На сегодняшний день, на наш взгляд, эта система наиболее совершенна с точки зрения решения технических задач и приспособлена к использованию вычислительной техники.

Что же такое ТРИЗ? Прообразом ТРИЗа является алгоритм решения изобретательской задачи, разработанный советским инженером Г. С. Альтшуллером в 1947 г. Этот метод позволяет активизировать творческий процесс, вести поиск решения технических задач более целенаправленно, познавая закономерности развития техники. Основой для этого служит богатый фонд изобретений, по существу история развития техники. Анализ более 40 тыс. изобретений показал, что, несмотря на огромное многообразие изобретательских задач, содержащиеся в них противоречия очень часто повторяются, т. е. существуют типовые противоречия, а следовательно, есть и типовые способы их преодоления. Было определено 40 наиболее эффективных приемов разрешения технических противоречий, объединенных в таблицы.

Кроме того, в системе имеются стандарты на решения изобретательских задач, т. е. определены средства, какими данные приемы могут быть осуществлены. В стандартах приводятся конкретные примеры, иллюстрирующие эти приемы.

В США была опубликована книга Альтшуллера, но так как она была научно-популярной, то ее не приняли всерьез. Кроме того, у американцев господствовало мнение, что если творческие задачи решает человек, то надо стараться повысить его психологический уровень, в частности имитируя метод мозгового штурма, предлагая достаточно хаотичный набор самых разных возможных решений. Но так повышается творческая активность лишь на проценты. Альтшуллер же ввел в свою теорию принципы эволюции технических систем, методы решений и даже психологию изобретательства. И тогда эффективность изобретательства повысилась в сотни раз.

В последние годы Г. С. Альтшуллер обратился к социальному патентному фонду и проанализировал 10 тыс. биографий выдающихся людей. Выяснилось, что в их жизни было очень много общего, и в результате анализа получилась схема, в которую укладывается практически любая личность. Удалось сформулировать приемы, дающие импульс к эффективному развитию творческого воображения и возможность в каждом человеке сформировать творческую личность.

Поскольку для решения изобретательских задач теорией Альтшуллера используется определенный алгоритм ведения поиска решения и, кроме того, имеется объемный информационный фонд, то стала очевидна необходимость привлечения вычислительной техники для решения задач ТРИЗ.

В итоге такой работы группой авторов во главе с В. М. Цуриковым из Минска был создан программный продукт «Изобретающая машина» (ИМ).

Интеллектуальная система ИМ создана на базе компьютера *IBM PC XT/AT*. Программа для изобретающей машины представляет собой систему последовательных, определенным образом составленных вопросов, конкретизирующих имеющиеся технические противоречия и подсказывающих способы их устранения (для этого в программу заложен богатый информационный фонд).

Использование интеллектуальной системы «Изобретающая машина» резко повысило интерес учащихся к изучению предмета «Основы технического творчества».

В сентябре 1991 г в техникуме создан хозрасчетный учебно-методический центр «Интеллектуальный сервис» (ИНТСервис) для ведения научно-исследовательской, производственной и коммерческой деятельности в интересах техникума путем создания, использования и реализации программного обеспечения, оказания интеллектуальных и иных услуг в деле развития творческих способностей, а также в области информатики и вычислительной техники.

С научно-исследовательской лабораторией изобретающих машин (НИЛИМ) из Минска (владельцем авторских прав на систему ИМ) техникум заключил договор о дилерстве, и центр ИНТСервис получил возможность реализации перечисленных ниже интеллектуальных систем:

ИМ-П (приемы) — генерирует идеи решений высокого уровня качества для 1250 типов изобретательских задач.

ИМ-С (стандарты) — дает гарантированно сильные решения сложных задач со структурным прогнозом развития полученной идеи.

ИМ-Э (эффекты) — содержит сотни рекомендаций по применению физических, химических, геометрических эффектов при решении изобретательских задач.

ИМ-ФСА — проводит функциональностоимостный анализ с целью снижения себестоимости и повышения качества выпускаемой продукции.

ИМ- (заявка) — предназначена для превращения изобретательской идеи в конечный продукт — комплект документов для подачи заявки на получение патента.

Все интеллектуальные системы семейства ИМ имеют дружественный интерфейс, высококачественные графические иллюстрации и обширные сервисные возможности.

Познакомиться и при желании приобрести ИМ вы можете на демонстрациях, проводимых ежедневно в ИНТСервисе.

Следует отметить также наличие альтернативных ИМ представительниц новейших компьютерных технологий для изобретательской деятельности, таких, например, как экспертные системы «Принцип действия» и «Альтернатива», распространяемые научно-техническим кооперативом «Метод» и реализуемыми на компьютерах *IBM PC XT/AT*.

Двухлетний опыт преподавания предмета «Основы технического творчества» позволяет сформулировать следующие выводы:

1. Введение предмета «Основы технического творчества» дает возможность объединить творческим началом все изучаемые на последующих курсах технические дисциплины.
2. Изучение предмета «Основы технического творчества» на I курсе позволяет быстро выявить учащихся с творческим воображением.
3. Эффективность изучения нового курса резко увеличивается за счет использования основ ТРИЗ и особенно интеллектуальной системы ИМ.
4. В то же время слабая подготовка учащихся I курса (ведь еще только изучается математика и физика в объеме средней школы) не позволяет получить большую эффективность от изучения предмета. Для устранения этого недостатка предлагается разработать программу курса «Основы профессионального творчества» для его непрерывного изучения с 1-го по 4-й годы обучения. Корректировка учебных планов позволила выделить следующие часы для изучения предмета ТРИЗ: I курс — 78 ч, II и III — по 30 ч, IV — 60 ч, причем знакомиться с ИМ учащиеся начинают с IV курса.

Коллектив техникума надеется, что все перечисленные выше мероприятия резко повысят эффективность обучения учащихся и будут способствовать формированию специалиста с оригинальным творческим воображением.

Н. ЛУНИНА

Краткое описание пакета учебных программ «Биология в школе»

В Научно-исследовательском вычислительном центре РАН разработан пакет учебных программ «Биология в школе». Существуют две версии пакета — для ЭВМ «Ямаха» и для *IBM PC*, имеющие незначительные различия. Версия для *IBM PC* содержит 10 программ, из которых 5 можно отнести к теме «Генетика» («Анализирующее скрещивание», «Получение чистых линий», «Взаимодействие генов», «Построение генетической карты», «Расшифровка генетического кода»), 3 — к теме «Биохимия» («Липиды», «Структурные формулы аминокислот», «Вторичная структура транспортных РНК») и 2 — к теме «Биосинтез белка» («Демонстрационная программа», «Биосинтез белка. Моделирующая программа»). Это деление в достаточной степени условно, поскольку нередко в программах затрагиваются вопросы из смежных тем.

Большинство программ построено по оди-

наковой схеме. В начале дается краткое изложение изучаемого вопроса и инструкция по работе с программой. Затем перед учеником ставится некоторая биологическая задача, для решения которой он должен провести эксперимент. Эксперимент моделируется. Например, от ученика требуется:

установить генотип самца кролика, скрещивая его с различными самками;

выделить чистые линии томатов,

построить бислой из случайно ориентированных фосфолипидов, расшифровать таблицу генетического кода, воздействуя на РНК мутагенными факторами и анализируя считываемый белок.

Данные для эксперимента задаются с использованием датчика случайных чисел, и поэтому два ученика за соседними машинами работают с разными параметрами (со своим полем томатов или со своей РНК). На экране обычно приводится перечень допустимых действий, а также указывается, как можно снять программу или запросить подсказку. При запросе подсказки первый раз даются наводящие соображения, второй раз (в этой же ситуации) сообщается правильный ответ, который ученик, тем не менее, должен сам ввести для продолжения работы программы. При работе с программами ведется счет, учитывающий правильные действия, ошибки, запросы, подсказки. В конце работы результаты (процент набранных очков от максимально возможных) вместе с фамилией ученика, датой и названием программы сохраняется в файле. Подробное описание программ содержится в работах [1—3]. В журнале «Биология в школе» предполагается также выход статей, посвященных программам по генетике и биохимии. Программы можно получить, обратившись по адресу: 142292 г. Пущино Московской области, НИВЦ РАН, Фонд алгоритмов и программ.

Литература

1. *Лунина Н. Л.* Компьютерная поддержка курса биологии // Биология в школе. 1988. № 5.

2. *Лунина Н. Л.* Учебные программы по биологии. Биосинтез белка: Материалы по математическому обеспечению ЭВМ / ОНТИ НЦБИ АН СССР. Пущино, 1988.

3. *Лунина Н. Л.* Учебные программы по биологии. Генетика: Материалы по математическому обеспечению ЭВМ / ОНТИ НЦБИ АН СССР. Пущино, 1990.

ЧТО МОЖЕТ ЭВМ?

Компьютеризация требует деликатности

Фирмы Microsoft, Summit Systems и некоторые другие объявили Луганск неблагоприятным для деловых контактов после того, как во время выставки «Компьютер 91», состоявшейся в этом городе, произошло множество криминальных эксцессов. В частности, был избит представитель Microsoft, а ОМОН не только не защитил «фирмачей», но даже препятствовал вызову представителей правоохранительных органов.

Кого что заботит

В Москве создается система сотовой телефонной связи (мобильные радиотелефоны), использующая диапазон 450 МГц. Она дороже и менее удобна, чем системы, использующие частоту 900 МГц (именно такие эксплуатируются почти во всем остальном мире), но диапазон 900 МГц в СНГ занят военными.

В то же время палата представителей конгресса США приняла законопроект об изъятии из ведения Пентагона полосы радиочастот шириной 200 МГц и использовании ее в гражданских целях.

Так может, американцы потому и богаче, что больше думают не «об воевать», а «об заработать»?

Если бы...

Фирма Cornestone Technology, производящая 19-дюймовые черно-белые мониторы (черные буквы на белом фоне) высокого разрешения с частотой кадров 67 Гц, заказала исследование их эффективности. Оказалось, что работоспособность оператора, использующего такой монитор, на треть выше, чем у работающего на VGA. Так что если в школах заменить VGA на Cornestone Technology, 10-летку можно было бы уложить в 7 лет.

Правда, сперва нужно дождаться, чтобы все школы были оснащены мониторами стандарта VGA (640×380 строк, не менее 16 цветов); все учебные программы существенно опирались на компьютерные технологии; все врачи не возражали против длительного сидения перед дисплеем...

Новая профессия дисковода

Еще совсем недавно у людей, имеющих дело с компьютерами, слово «дисковод» вызывало мышечное напряжение. Еще бы, ведь это устройство размером со стиральную машину заставляло операторов крайне осторожно устанавливать на его ось пакеты магнитных дисков весом в несколько килограммов. Сегодня жесткие диски такой же емкости имеют размер карманного томика, ну а появление лазерных дисков окончательно сняло проблему физического развития пользователей.

Но если для программистов смена дисков носит творческий характер, то поклонников цифровой дисковой музыки частенько раздражает поиск и установка любимых хитов в лазерный проигрыватель.

Чтобы избавить их от рутинной работы, на фирме Pioneer Electronic было создано устройство для автоматической смены лазерных дисков. Нужная последовательность установки любого из 300 дисков и выбор необходимых фонограмм на них программируется заранее и может быть изменена в любой момент. Самая длинная пауза, необходимая для смены дисков, не превышает 5 с. Последовательность выбора дисков и фонограмм на них может быть любой.

И даже если вам очень нравится только одна песня, вы можете слушать ее снова и снова, пока кто-нибудь менее музыкальный не сбросит вашу программу кнопкой Reset.

Еще не записные книжки, но...

Большая группа компьютеров на американском рынке «перевалила» через барьер следующих показателей: вес менее 3,9 кг при наличии жесткого диска емкостью не менее 80 Мбайт, высококачественного дисплея «черное на белом», 4 Мбайт ОЗУ; цена не выше 5000 долларов.

На наш рынок фирма Sharp выставила notebook-компьютер PC-6240 с винчестером 40 Мбайт, дисплеем, имеющим разрешение 640×480 точек (16 оттенков серого) и 1 Мбайт ОЗУ; вес — 2 кг, толщина — 34 мм.

Ну, а на японском появился новый notebook (почему-то применительно к компьютерам это все чаще переводится на русский как «записная книжка», хотя размеры большинства представителей этого класса больше соответствуют термину «блокнот») фирмы Mitsubishi Electric. Он упаковывается в довольно большой кейс — 47×34×13 см, зато в комплект входят не только принтер, но и радиомодем с рацией, позволяющие включаться в информационную систему MCA Service Information System прямо «на ходу».

Multimedia — дело будущего

Ближайшего будущего, утверждают оптимистичные американцы. Обучение с одновременным использованием звука, движущегося изображения, баз данных с гиперсвязями — это фантастика! Но чтобы реализовать большие возможности, нужны большой труд и большие деньги, поэтому multimedia-курсов мало, и ориентированы они в первую очередь на обучение в сфере бизнеса, да и глубина использования возможностей оставляет, как говорится, желать. Например, «при обучении работе с электронной таблицей некоторые цифры на экране могут начать вращаться. Это означает, что вскоре они должны смениться другими» (М. Френч, руководительница группы коммерческих изделий отделения Entry Business Unit фирмы Microsoft об одном из курсов фирмы). Очень мило, но для чего здесь multimedia?

Всем, желающим идти в ногу со временем!

Совместное советско-канадское предприятие

«МДМ-Украина»,

учредителями которого являются Главный вычислительный центр Министерства народного образования Украины и канадская фирма MDMT Group Inc.,
предлагает вашему вниманию программные средства отраслевого фонда алгоритмов и программ, распространяемые нами по ценам ниже рыночных!

КУВТ «Корвет»

Пакет педагогических программ по курсу «Основы информатики и вычислительной техники».

Курс по изучению грамматики и лексики английского языка.

Локальная вычислительная сеть RDOS.

Программа вывода графической информации на печатающее устройство «GRAFPRN».

Практикум для медицинских училищ «Медик».

Комплекс учебных программ «Эмулятор языка ЛОГО».

Пакет по алгебре и началам анализа.

Компиляторы и библиотеки сетевых и графических функций для систем программирования ассемблер, Си, Паскаль.

Комплект музыкальных редакторов.

Компьютерные игры.

«Yamaha» MSX-1, MSX-2

Пакет по физической географии.

Пакет по алгебре и началам анализа.

Пакет по высшей математике.

Пакет по немецкому языку.

Пакет по биологии «Растительная клетка».

Пакет педагогических программ по курсу «Основы информатики и вычислительной техники».

Комплекс программ психодиагностики, дает возможность создавать интегративные личностные портреты, описывать и сравнивать личностные особенности и социальные стереотипы как в структуре малых групп и коллективов, так и на уровне личности как элемента группы.

Обучающий игровой пакет «Бука».

Пакет по физике.

Пакет педагогических учебно-контролирующих программ по Бейсику.

IBM PC XT/AT

Система контроля исполнения поручений КИПР.

Диалоговая система психофизиологического тестирования ДИСПОС.

Диалоговая система профессионального обучения ДИСПО.

Система моделирования общего назначения GPSS/PC V. 2.0.

Программный комплекс «Учет хозяйственной деятельности организации» включает АРМы учета: труда и зарплаты, основных средств, материальных ценностей, накладных расходов, фактических затрат, смет, кадров. АРМы охватывают деятельность планового отдела, бухгалтерии, отдела кадров и функционируют на общей базе данных.

Пакет по физической географии.

Курс по изучению грамматики и лексики английского языка.

Комплекс программ психодиагностики.

И другие программные средства, предлагаемые в каталоге, значительно облегчат вашу работу, помогут резко поднять методический и технический уровень использования терминальных классов.

Спешите приобрести каталог (объем 82 страницы с аннотациями), который высылается бесплатно!

Наш адрес: 254053, Киев-53, ул. Артема, 52-Д.
Для писем: 254053, Киев, а/я 33, Фонд.
Контактные телефоны: 213-97-77, 211-97-32.

Вниманию пользователей ПЭВМ RS/2, «Агат», «Электроника МС 0202» (УКНЦ), БК-0010

Чебоксарское научно-техническое предприятие «Логосъ», занятое развитием новых информационных технологий обучения, извещает об информационных услугах на 1992 г.

Разовое обслуживание

Мы предлагаем вам широкий выбор отлично зарекомендовавших себя педагогических программных средств, включающих программную поддержку школьных курсов информатики, математики, физики, химии, истории, биологии и других предметов; математическое моделирование естественных и гуманитарных явлений; новые прикладные, системные и инструментальные средства, графические игры.

Наши цены весьма умеренны: от 5 до 500 рублей. Мы с удовольствием вышлем вам подробный каталог программ бесплатно.

Мы сотрудничаем с ведущими заводами-изготовителями ВТ в СССР и поможем вам улучшить техническое состояние вашего класса, отремонтировав и модифицировав технику, установив сетевое оборудование, вы сможете приобрести у нас все необходимое: мониторы, дисковые накопители, принтеры, дискеты и пр.

Информационное обслуживание

Вы можете оформить годовое информационное обслуживание «Логосъ-информатор». Подписчиков ждут оригинальные коммерческие продукты предприятия «Логосъ», педагогические программные средства по различным школьным предметам, новые системные и прикладные программы, рекламные продукты, новые версии уже известных на рынке программ.

6 выпусков в год на 12 дискетах. Стоимость подписки на 20 % ниже, чем суммарная стоимость входящих в нее программных продуктов.

Подписка может быть оформлена в течение всего 1992 г. Стоимость подписки — 1500 рублей для ПЭВМ PS/2 и 1000 рублей для остальных ПЭВМ.

Условно бесплатное информационное обслуживание

Для частных лиц и организаций мы расширяем самую популярную подписку прошлого года, подписку на условно бесплатное годовое информационное обслуживание из фонда материалов, любезно переданных нам для некоммерческого использования. Содержание фонда интересно и многогранно: инструментальные средства, педагогические разработки, авторские программные продукты. Вы получите эти материалы (12 дискет за год), оплатив только стоимость дискет и накладные расходы.

Подписка может быть оформлена в течение всего 1992 г. Стоимость подписки — 600 рублей для ПЭВМ PS/2 и 200 рублей для остальных ПЭВМ.

Оформление заказа

Для оформления подписки на любой вид обслуживания следует направить в наш адрес письмо-заказ, указав в нем вид выбранного обслуживания, название вашей ПЭВМ, тип используемого на ней дисковода, точный почтовый адрес и банковские реквизиты. Оформление будет значительно ускорено, если в это же письмо будет вложена копия платежного поручения о переводе суммы подписки на наш счет.

Работа с авторами

Предприятие заключает договора с авторами-разработчиками программных продуктов для указанных ПЭВМ на тиражирование. Гарантируется возврат присланных магнитных дискет и соблюдение авторских прав.

Адрес для заказов и справок: 428034, г. Чебоксары, а/я 99, НМВМП «Логосъ». Тел. 42-82-77, 22-82-18. Р/с 609201 в Коммерческом банке «Чувашкредитпромбанк» г. Чебоксары. МФО 286352.

А. ГРУДЕВ

Задача о роботе

В качестве конкурсной задачи на первый тур IV Всесоюзной олимпиады школьников по информатике членом-корреспондентом РАН Ф. Л. Черноусько и мной была предложена и одобрена специальной методической комиссией следующая задача.

Задача «Робот»

На координатной плоскости задан робот-манипулятор, состоящий из двух звеньев длиной *L1* и *L2*. Первое звено шарнирно закреплено в начале координат, второе шарнирно соединено с первым. Робот управляется путем поворота звеньев в первом и втором шарнирах с угловыми скоростями, по абсолютной величине не превышающими *W1* и *W2* соответственно.

Робот предназначен для обработки деталей, расположенных в точках (x_1, y_1), (x_2, y_2), ..., (x_N, y_N).

В начальный момент первое звено лежит на оси *OX*, а второе звено образует с ней угол *Q*, отсчитываемый от оси *OX* против часовой стрелки. Время обработки каждой детали фиксированно и равно *P*.

Требуется составить программу, которая:

а) выдает какую-либо последовательность обработки деталей и требуемое для этого время;

б) выдает последовательность обработки деталей за минимальное по возможности время;

в) для заданного интервала времени [0, *T*] выбирает максимальное количество обработанных деталей.

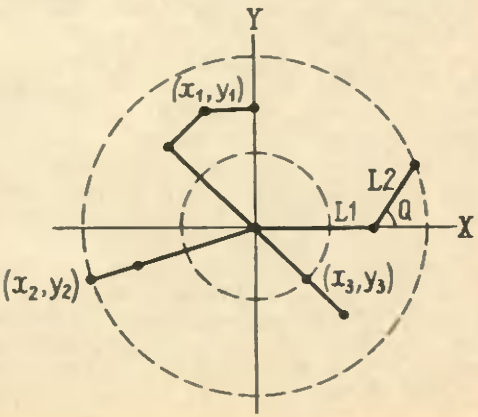
Примечания. 1. Все исходные данные целочисленные.

2. Программа должна осуществлять ввод данных либо из входного файла (*A:/ROBOT.DAT*), либо с клавиатуры в виде последовательности: *L1*, *L2*, *W1*, *W2*, *Q*, *N*, *T*, *P*, x_1, y_1, ..., x_N, y_N.

Предварительные замечания. Всякий достаточно квалифицированный и уважающий себя программист (а именно таковых мы ожидали увидеть на Всесоюзной олимпиаде по информатике среди школьников IX—XI классов), прочитав условия этой задачи, должен презрительно хмыкнуть: «Так это же полный перебор...» Мы не будем отпираться и честно признаемся, что представленная задача действительно относится к классу *NP*-полных задач [1], *точное* решение которых может быть найдено только с помощью полного перебора. Однако на этом очевидность решения (в алгоритмическом и программном смысле) заканчивается и начинается детальный, быть может несколько скучноватый, но весьма полезный процесс решения.

Разбор задачи. Сначала мы подробно рассмотрим существо задачи и напишем некоторые формулы; затем построим и опишем алгоритм решения; потом приведем текст программы и дополним его необходимыми комментариями и только в самом конце уделим немного места анализу типичных ошибок, допущенных школьниками, и дадим несколько полезных советов, дабы избежать по возможности их повторения.

Итак, о чем же идет речь? На рис. 1 схематично изображен двухзвенный робот-манипулятор. Каждое звено вращается относительно своего конца независимо от того, вращается или находится в покое другое



звено. В условии ничего не сказано о недопустимости пересечения звеньев (раз не сказано — значит, можно!). Следовательно, теоретически (да и не только) можно и нужно предусмотреть возможность обработки деталей в случаях, когда звенья робота перекрываются. Такое возможно, например, когда $L_1=L_2$ и $x_i=y_i=0$.

Прежде чем переходить к описанию математической модели, остановимся на двух существенных моментах — особенностях конструкции. Рассмотрим i-ую деталь на плоскости, находящуюся в точке с координатами (x_i, y_i). Зададимся вопросом (с чего бы это?): всегда ли может робот обработать деталь, т. е. любая ли точка достижима? Уже по вопросу понятно: нет, не любая. Действительно, расстояние от начала координат до точки равняется r_i:

$$r_i=\sqrt{x_i^2+y_i^2}. \qquad (1)$$

Если же суммарная длина звеньев меньше этого расстояния, т. е. если $(L_1+L_2)<r_i$, то соответствующая деталь останется недостижимой, поскольку робот просто «не дотянется» до нее. Таким образом, существует «внешняя зона недоступности», границей которой является окружность радиуса (L_1+L_2).

Не столь очевиден, но так же прост другой случай, когда звенья имеют разную длину $(L_1 \neq L_2)$ и вследствие этого имеется «внутренняя зона недоступности» вокруг начала координат. Радиус этой зоны равен разнице в длинах звеньев. А это означает, что если $|L_1-L_2|>r_i$, то i-я деталь также недоступна.

Границы обеих «зон недоступности» изображены на рис. 1 пунктирными окружностями. Итак, только те детали, которые расположены внутри или на границах «кольца достижимости», ограниченного окружностями радиуса $|L_1-L_2|$ и (L_1+L_2), могут быть обработаны роботом.

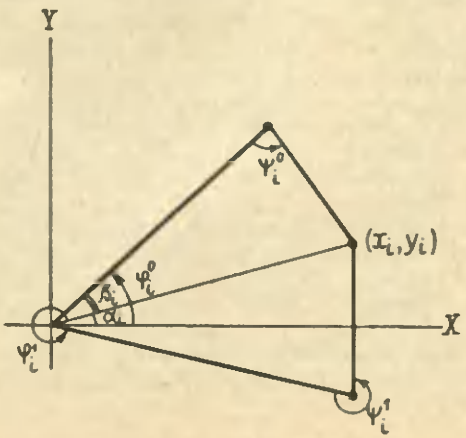
Теперь зададимся еще более каверзным вопросом: а сколькими путями может быть достигнута каждая деталь? И опять же из постановки вопроса ясно: по крайней мере не одним. А сколькими же? Ответ таков: одним или двумя за единственным исключением. На рис. 1 изображены конфигурации робота в моменты достижения им трех деталей. Первая деталь может быть обработана роботом в любой из двух конфигураций. Отрезок, соединяющий начало координат с точкой (x_i, y_i), и два звена робота (постоянной длины) образуют треугольники. Хорошо известно, что из трех отрезков можно составить не более двух различных треугольников (в том, что они действительно различны, можно убедиться, попытавшись совместить их, не выходя из плоскости рисунка). Следовательно, только двумя способами может быть достигнута каждая деталь, находящаяся *внутри* «кольца достижимости». Если звенья робота располагаются слева от отрезка, соединяющего точку (0, 0) с точкой (x_i, y_i), то будем называть это левой конфигурацией робота. В противном случае будем говорить о правой конфигурации.

При обработке детали, находящейся в точке (x_2, y_2), звенья робота вытягиваются в одну линию, поскольку эта точка лежит на внешней границе «кольца достижимости». Следовательно, оба из вышеупомянутых треугольников вырождаются в один отрезок и в этом случае говорить о левой и правой конфигурациях, вообще говоря, не приходится.

То же самое относится к точке (x_3, y_3), лежащей на *внутренней границе* «кольца достижимости». Однако для упрощения описания алгоритма (а также написания программы) будем считать, что и в этих двух последних случаях существуют две конфигурации робота — левая и правая, правда совпадающие.

Все это было бы так, если бы не единственное исключение, доставляющее массу хлопот и неприятностей. Речь идет о точке с координатами (0, 0) — начале координат. Дело в том, что робот может достигнуть этой точки, находясь в бесконечно большом числе конфигураций: при этом звенья сложены (перекрывают друг друга полностью) и конец второго звена, называемый обычно схватом, располагается над (или под) основанием первого звена, т. е. угол между звеньями при этом равен 0. Положение же первого звена на плоскости неопределенно — любое положение пригодно для обработки детали, находящейся в начале координат.

Немного математики. Итак, положение робота, как это видно на рис. 2, в общем случае определяется двумя углами (при этом



механики говорят, что система с двумя степенями свободы описывается двумя обобщенными координатами): φ и Ψ. Здесь φ — угол между положительным направлением оси OX и первым звеном, отсчитываемый в положительном направлении, т. е. против часовой стрелки. По определению ясно, что $\varphi \in [0, 2\pi[$. Через Ψ обозначен угол между звеньями, отсчитываемый от первого звена до второго в положительном направлении и лежащий также в пределах $[0, 2\pi[$.

Если мы будем говорить о положении робота (или о его конфигурации), соответствующем некоторой i-й точке (разумеется, доступной!), то следует различать положения, отвечающие левой и правой конфигурациям. Обозначим через $\{\varphi_i^0, \Psi_i^0\}$ левую конфигурацию, соответствующую i-й точке, а через $\{\varphi_i^1, \Psi_i^1\}$ правую конфигурацию робота, достигнувшего этой же точки.

Случай $i=0$ будет соответствовать начальному положению робота, и для общности обозначений будем считать, что $\varphi_0^1=\varphi_0^0$ и $\Psi_0^1=\Psi_0^0$. Из условия задачи следует, что

$$\varphi_0^1=\varphi_0^2=0,\ \Psi_0^1=\Psi_0^0=\pi+Q. \qquad (2)$$

Для точек, лежащих на границе зоны достижимости, очевидно, что $\varphi_i^1=\varphi_i^0$ и $\Psi_i^1=\Psi_i^0$. Что же касается точки $(0, 0)$, то, к сожалению, только про угол Ψ известно, что $\Psi^1=\Psi^0=0$, а про угол φ ничего не ясно и любое его значение из области определения $[0, 2\pi[$ может быть вполне пригодно.

Итак, первый шаг сделан: от железного робота мы перешли к набору из множеств $\{\varphi_i^0, \Psi_i^0\}$ и $\{\varphi_i^1, \Psi_i^1\}$. Осталось только выяснить, как же эти углы подсчитать. На этот вопрос дает ответ несложная планиметрия с использованием элементов тригонометрии (теоремы косинусов). Для простоты изложения будем рассматривать i-ю точку в первом квадранте. Введя, как это видно на рис. 2, дополнительные углы α_i и β_i, получаем:

$$\varphi_i^0=\alpha_i+\beta_i,\ \varphi_i^1=\alpha_i-\beta_i,\ \Psi_i^1=2\pi-\Psi_i^0. \qquad (3)$$

Величина α_i находится из рассмотрения прямоугольного треугольника:

$$\alpha_i = arctg \frac{y_i}{x_i}. \qquad (4)$$

Величины β_i и Ψ_i^0 находятся из рассмотрения треугольника, образованного звеньями робота:

$$\beta_i = arccos \frac{(L_1)^2+(r_i)^2-(L_2)^2}{2L_1r_i}, \qquad (5)$$

$$\Psi_i = arccos \frac{(L_1)^2+(L_2)^2-(r_i)^2}{2L_1L_2}. \qquad (6)$$

Подстановка выражений (4), (5) и (6) в формулы (3) дает два множества: $\{\varphi_i^0, \Psi_i^0\}$ и $\{\varphi_i^1, \Psi_i^1\}$.

Искомая величина — T_{min} — минимальное время обхода всех деталей, очевидно, представляет собой сумму времен t_{ij}, где t_{ij} — время перехода от i-й детали к j-й плюс время обработки j-й детали. Однако здесь нам не обойтись двумя индексами i и j. Существенную роль при вычислении времени перехода от одной детали к другой играет ответ на вопрос: из *какой* конфигурации (левой или правой), соответствующей i-й точке, в *какую* конфигурацию (правую или левую), соответствующую j-й точке, переходит робот? Условимся поэтому обозначать время перехода из i-й точки в j-ю точку через t_{ij} с двумя дополнительными верхними индексами, первый из которых соответствует конфигурации робота при обработке i-й точки, а второй — для j-й точки. Так, например, t_{ij}^{10} обозначает переход из правой конфигурации (верхний индекс 1) i-й точки в левую конфигурацию (верхний индекс 0) j-й точки.

Сразу же сделаем следующее, достаточно очевидное замечание, которое, однако, потребует определенных усилий при программной реализации. Нет смысла вычислять значения времен t_{ij} при $i=j$, поскольку переход из некоторой точки в нее же саму суммарного времени обхода не уменьшит. Таким образом, в общем случае переход из i-й точки в j-ю сопряжен с вычислением не одного, а четырех времен: t_{ij}^{00}, t_{ij}^{01}, t_{ij}^{10}, t_{ij}^{11}.

Как же вычисляются эти времена? Продемонстрируем это на примере вычисления времени t_{ij}^{10}. Понятно, что при переходе от одной точки к другой независимо могут изменяться углы, определяющие конфигурацию робота:

$$\varphi_i^1 \Rightarrow \varphi_j^0,\ \Psi_i^1 \Rightarrow \Psi_j^0.$$

На поворот каждого из звеньев требуется некоторое время. Так, на поворот первого звена на угол

$$\Delta\varphi^{10}=\varphi_i^1-\varphi_j^0 \qquad (7)$$

требуется время, которое находится из формулы:

$$W_1 t_{ij}^{10}=|\Delta\varphi_{ij}^{10}|, \qquad (8)$$

где произведение угловой скорости на время поворота равняется изменению угла. Знак модуля в правой части стоит вследствие того, что W_1 по смыслу задачи означает абсолютную величину скорости независимо от направления вращения. Однако уже здесь следует обратить внимание на то, что углы φ и Ψ определены нами только на интервале $[0, 2\pi[$. Поэтому изменение угла φ от зна-

чения $\pi/10$, например, до значения $19\pi/10$ составляет всего лишь $2\pi/10$, а не $18\pi/10$, как это следует из выражения (7). Поэтому вместо выражения (7), с учетом области определения углов φ и Ψ, следует использовать более точное выражение:

$$|\Delta\varphi_{ij}^{lo}| = \min\{|\varphi_j^l - \varphi_i^o - 2\pi|; |\varphi_j^l - \varphi_i^o|; |\varphi_j^l - \varphi_i^o + 2\pi|\}. \quad (9)$$

Подставляя выражение (9) в формулу (8), получаем значение t_{1ij}^{lo}. Аналогично определяется время поворота второго звена (в общем случае из i-й точки с k-й конфигурацией в j-ю точку с l-й конфигурацией):

$$t_{2ij}^{kl} = \Delta\Psi_{ij}^{kl}/W_2,$$
$$\Delta\Psi_{ij}^{kl} = \min\{|\Psi_i^k - \Psi_j^l - 2\pi|; |\Psi_i^k - \Psi_j^l|; |\Psi_i^k - \Psi_j^l + 2\pi|\}. \quad (10)$$

Теперь мы знаем t_{1ij}^{kl} и t_{2ij}^{kl} — времена поворотов звеньев при переходе из i-й точки в j-ю. Ясно, что время разворота робота из i-й точки в j-ю равняется максимальному из этих двух времен:

$$t_{ij}^{kl} = \max\{t_{1ij}^{kl}, t_{2ij}^{kl}\}. \quad (11)$$

Итак, вычислив t_{ij}^{kl} при всевозможных $k,l \in \{0, 1\}$, мы получили матрицу (массив) $T = \{t_{ij}^{kl}\}_{k,l \in \{0,1\}; i,j \in \{0,n\}}$. Здесь мы впервые использовали переменную n для обозначения числа *доступных* точек, которое может отличаться от полного числа N вводимых точек. Таким образом, мы свели искомую задачу к задаче коммивояжера [1], в которой элементами матрицы также являются времена перехода из одного пункта в другой.

Несколько замечаний относительно структуры матрицы. Интуитивно ясно, что время перехода из i-й точки в j-ю равно времени обратного перехода; из j-й точки в i-ю (это все верно для каждой из конфигураций), т. е.

$$t_{ij}^{kl} = t_{ji}^{lk}. \quad (12)$$

Это означает, что матрица T симметрична относительно главной диагонали. Кстати говоря, элементы главной диагонали не требуют вычисления в силу сделанного выше замечания.

А сколько же элементов содержит матрица T? Ответ прост:

$$|2(n+1) \times 2(n+1)| = 4(n+1)^2.$$

Описание алгоритма. Для того чтобы определить, какая же последовательность точек обработки дает минимальное время, нужно, вообще говоря, просматривать всевозможные *перестановки* порядка обработки деталей. Обозначим через a_i номер точки (полученный ею при вводе данных), которая должна быть обработана в i-ю очередь. Так, например, если $a_3 = 7$, то третьей по счету (после начальной) будет обработана седьмая точка. Напоминаем, что нас интересуют последовательности всех n точек, не содержащие повторяющихся. Иными словами, если первая тестируемая последовательность есть $A_1 = \{a_i\}_{i=1}^n$, где $a_i = l$ при $l \in [1, n]$, то нас интересуют всевозможные перестановки этих n чисел (a_i).

Прежде чем перейти к подробному описанию алгоритма решения задачи, назовем метод ее решения — метод ветвей и границ. Этот метод является наиболее доступным и продуктивным при решении *NP*-полных задач. Он заключается в том, что в процессе поиска цепочки (последовательности точек) с минимальным временем не все цепочки следует просматривать до конца. Например, если часть некоторой цепочки B дает время, большее, чем T_{min} полученное в процессе прохода *всей* другой, ранее проверенной цепочки A, то далее цепочка B не просматривается.

Сколько же цепочек предстоит просмотреть, прежде чем мы с уверенностью скажем: вот такой порядок обработки деталей дает наименьшее время обработки T_{min}? Ответ на этот вопрос хорошо известен из комбинаторики: $n!$ (не восклицательный знак, а знак факториала: $n! = 1 \cdot 2 \cdot 3 \cdot \ldots \cdot (n-1) \cdot n$). К сожалению, этот ответ не совсем, мягко говоря, верен. Дело в том, что при некоторой фиксированной последовательности обработки деталей A^* мы можем выбрать еще 2^n различных путей. Вспомним, что каждой доступной точки можно достичь в общем случае двумя путями. Поэтому мы каждой последовательности A^* сопоставим 2^n множеств $S^* = \{s_1, \ldots, s_n\}$. Множество S представляет собой n цифр (0 или 1), каждая из которых обозначает конфигурацию в соответствующей точке. Так, например, если $a_3 = 7$ и $s_3 = 0$, то седьмая точка будет обработана третьей по счету в левой конфигурации.

Теперь мы знаем точное количество цепочек с учетом конфигураций: $2^n n!$ — это так называемая *вычислительная сложность* нашей задачи. Однако метод ветвей и границ в каждом конкретном случае позволяет просматривать меньшее число цепочек (попробуйте приведенную ниже программу модифицировать таким образом, чтобы получать число *полностью* просмотренных цепочек в каждом конкретном случае).

Настала пора решить вопрос о том, как же подсчитывать время перехода через (0, 0), иными словами, какую же конфигурацию робота приписать случаю, когда схват находится в точке (0, 0). Ответ будет таким. Пусть i^* — номер детали, находящейся

в начале координат. Тогда $\Psi^{0}_{i^*}=\Psi^{l}_{i^*}=0$. А угол φ_{i^*} будет различным в зависимости от того, в какую точку робот проследует *после* точки i^*. А как же быть с матрицей T? — спросите вы. И будете правы: в случае, если среди точек имеется *допустимая* точка $(0, 0)$, некоторые элементы матрицы T *не будут* вычисляться (а именно: $t^{kl}_{i^*j}$ при всех $j\in[0, n]$ и $k, l\in\{0, 1\}$). Но суммарное время T_{min} как-то нужно вычислять, и мы поступим следующим образом. Пусть при переходе от точки (i^*-1) до точки i^* второму звену нужно время $t^{kl}_{(i^*-1)i^*}$ для того, чтобы угол $\varphi_{i^*}=0$. Кстати сказать, это время будет одинаковым для всех $k, l\in\{0, 1\}$. Поскольку звенья вращаются независимо, то будем считать, что все это время $t_{(i^*-1)i^*}$ первое звено вращается так, как будто после точки (i^*-1) следует точка (i^*+1), т. е. как бы опуская точку $(0, 0)$. На достижимости точки $(0, 0)$ это никак не скажется, а вот при переходе из точки $(0, 0)$ в точку (i^*+1) будет получено минимальное время, определяемое либо временем поворота второго звена ($t^{kl}_{i^*(i^*+1)}$), либо временем на выработку *остатка* угла ($\varphi^{m}_{i^*+1}-\varphi^{k}_{i^*-1}$). При этом мы предполагаем, что во время обработки детали первое звено поворачиваться не будет. Можно доказать, что в результате всех этих ухищрений будет получено минимальное время на участке от (i^*-1)-й до (i^*+1)-й точки.

Теперь предлагается следующий алгоритм решения задачи:

1. Ввод данных с контролем.

2. Ввод координат точек. Проверка каждой точки на предмет принадлежности ее к области достижимости. Запоминание порядкового номера подходящей точки и вычисление для нее двух конфигураций (т. е. четырех углов) робота. Если допустимая точка является началом координат, то запоминаем ее порядковый номер.

3. Формирование матрицы T, т. е. вычисление времен t^{kl}_{ij} для всех $ij\in[0, n]$ и $k, l\in\{0, 1\}$ с учетом вышесказанного об обработке точки $(0, 0)$.

4. Поочередная генерация последовательностей обхода точек, начиная с A_1. Для каждой последовательности формируется 2^n множеств S. Для каждого множества конфигураций подсчитывается время обработки каждой детали. Попутно подсчитывается количество деталей, обработанных в течение заданного интервала времени $[0, T]$, и если это количество превышает ранее полученное, то фиксируется новое значение (таким образом, «за один проход» решаются подзадачи б и в). Если после прохода *всех n* точек выясняется, что время обхода меньше полученного раньше, то это время запоминается и фиксируется соответствующая ему последовательность A обхода точек. Если в процессе прохода точек выясняется, что: 1) время обработки первых j деталей из данной последовательности A' при данном множестве S' *превышает* полученное ранее и 2) количество обработанных деталей *меньше* ранее полученного числа деталей, обработанных за фиксированный интервал времени $[0, T]$, то дальнейший просмотр деталей прекращается и формируется новое множество, игнорируя *всевозможные* множества, имеющие с множеством S' одинаковые первые $(j-1)$ цифр. Иначе говоря, отбрасываются все «ветви» на графе, исходящие от j-го узла,— это и есть реализация метода ветвей и границ.

5. Печатается минимальное время обхода n точек и соответствующая ей последовательность этих точек.

6. Печатается максимальное число деталей, обработанных в течение заданного интервала времени $[0, T]$.

Заметим, что пункт 5 алгоритма дает ответ сразу на два пункта задачи: а и б. Отметим также, что данный алгоритм не является оптимальным с точки зрения времени работы программы. Нами создан и оптимальный по быстродействию алгоритм, в котором учитывается, что в точках, лежащих на *границе* области достижимости, робот принимает единственную конфигурацию. Это снижает число множеств S, соответствующих некоторой последовательности обхода точек. Так, например, для рис. 1 каждой последовательности A будет соответствовать не $2^3=8$, а всего лишь два множества. Однако программа, реализующая оптимальный алгоритм, становится в полтора раза больше в объеме.

Дальше мы приводим текст нашей программы, которая полностью лишена комментариев. Мы знаем, что это дурной тон, но делаем это в целях экономии места. Необходимые и более подробные комментарии к программе мы приведем ниже.

```
1 #include <math.h>
2 #define PI 3.141593
3 #define swap(x,y) {(x)=(x)+(y);(y)=(x)-(y);(x)=(x)-(y);}
4 #define NORM(x) ((((x)>=0)&&((x)<=2*PI))? (x):(((x)<0)? (x)+2*PI:(x)-2*PI))
5 #define HEVIS(x) (((x)>0)? 1:0)
6 #define ABS(x) (((x)<0)? -(x):(x))
```

```
#define conf(i,j,k) cv[(i)*4+(j)*2+(k)]
#define tim(i,j,k,l) tv[(i)*(n+1)*4+(j)*4+(k)*2+(l)]
 float *cv;

   void main()
  {
   int L1,L2,W[2],Fi0,Np,T_fix,T_det;
   int x,y,fl1,i,j,k,l,m,ij,n,*or,*posl,*num,*seq,fl,kod=1,kod1=1,N_max=0;
   float t,t1,T_min=1E20,l1,alfa,beta,dt[2],*tv,FIMIN();
   do
     { printf("\nВведите длины звеньев  L1 и L2 ((L1+L2)>0)\n");
       scanf("%d%d",&L1,&L2);
     }
   while((L1==0)&&(L2==0));
   do
     { printf("Введите угловые скорости W1 и W2 ((W1+W2)>0)\n");
       scanf("%d%d",&W[0],&W[1]);
     }
   while((W[0]<0)&&(W[1]<0));
   printf("Введите начальный угол Q в радианах\n");
   scanf("%d",&Fi0);
   do
     { printf("Введите число деталей N>0\n");
       scanf("%d",&Np);
     }
   while(Np<=0);
   do
     { printf("Введите время T>0\n");
       scanf("%d",&T_fix);
     }
   while(T_fix<=0);
   do
     { printf("Введите время обработки детали P>0\n");
       scanf("%d",&T_det);
     }
   while(T_det<=0);
   cv=malloc(4*sizeof(float));
   num=calloc(1,sizeof(int));
   conf(0,0,0)=conf(0,1,0)=0;
   conf(0,0,1)=conf(0,1,1)=NORM(PI+Fi0);
   for (i=1,n=0;i<=Np;i++)
      { printf("Введите координаты  X,Y %d-ой точки\n",i);
        scanf("%d%d",&x,&y);
        if (((x*x+y*y)<=((L1+L2)*(L1+L2)))&&((x*x+y*y)>=(L1-L2)*(L1-L2)))
           { cv=realloc(cv,4*(++n+1)*sizeof(float));
             num=realloc(num,4*(n+1)*sizeof(float));
             num[n]=i;
             alfa=(x==0)? PI/2+HEVIS(-y)*PI:NORM(atan(y/x)+HEVIS(-x)*PI);
             l1=x*x+y*y;
             beta=(l1==0)? 0:acos((L1*L1+l1-L2*L2)/(2*L1*sqrt(l1)));
             conf(n,0,0)=NORM(alfa+beta);
             conf(n,0,1)=acos((L1*L1+L2*L2-l1)/(2*L1*L2));
             conf(n,1,0)=NORM(alfa-beta);
             conf(n,1,1)=2*PI-conf(n,0,1);
             fl=(l1==0)? n:-1;
           }
      }
   tv=calloc(4*(n+1)*(n+1),sizeof(float));
   seq=malloc((n+1)*sizeof(int));
   posl=malloc((n+1)*sizeof(int));
   or=calloc(n+1,sizeof(int));
```

```
  for (i=1;i<=n;i++)
     for (ij=-1,j=0;j<=n-1;j++)
        if (i>j)
         { if ((i==fl)||(j==fl))
             ij=(i==fl)? j:i;
           if (ij>-1)
tim(ij,i+j-ij,0,0)=tim(ij,i+j-ij,0,1)=tim(ij,i+j-ij,1,0)=tim(ij,i+j-ij,1,1)=
ABS(2*PI*HEVIS(conf(ij,0,1)-PI)-conf(ij,0,1))/W[1];
           else
           {
             for (k=0;k<=1;k++)
                for (l=0;l<=1;l++)
                   { for (m=0;m<=1;m++)
                      dt[m]=FIMIN(i,j,k,l,m)/W[m];
                      tim(j,i,l,k)=tim(i,j,k,l)=(dt[0]>dt[1])? dt[0]:dt[1];
                   }
           }
        }
  for (i=0;i<=n;i++)
     seq[i]=i;
  while(kod)
     { while(kod1)
         { for (t=0,i=fl1=1;(i<=n)&&(fl1);i++)
             { if (seq[i-1]==fl)
                 { t1=(FIMIN(seq[i],seq[i-2],or[i],or[i-2],0))/W[0]-
                     tim(seq[i-2],seq[i-1],or[i-2],or[i-1]);
                   t+=T_det+((tim(seq[i],seq[i-1],0,0)>t1)?
                      tim(seq[i],seq[i-1],0,0):t1);
                 }
               else t+=tim(seq[i-1],seq[i],or[i-1],or[i])+T_det;
               if (t<=T_fix)
                 N_max=(i>N_max)? i:N_max;
               else fl1=HEVIS(T_min-t);
             }
           if (t<T_min)
             for (T_min=t,k=0;k<=n;k++)
               posl[k]=seq[k];
           for (j=i-1;(j>0)&&(or[j]==1);j--);
           kod1=or[j]=HEVIS(j);
           for (k=j+1;k<=n;k++)
              or[k]=0;
         }
         for (i=n;(seq[i-1]>seq[i])&&(i>1);i--);
         if (i!=1)
           { for (j=n;seq[j]<seq[i-1];j--);
               swap(seq[j],seq[i-1]);
               for (k=(n-i+1)/2,j=1;j<=k;j++)
                  swap(seq[n-j+1],seq[i+j-1]);
           }
         kod1=kod=HEVIS(i-1);
     }
  printf("\nT_min=%f",T_min);
  printf("\nСоответствующая последовательность деталей:");
  for (i=0;i<=n;i++)
     printf("\n %d",num[posl[i]]);
  printf("\nМаксимальное число обработанных деталей:%d",N_max);
 }

float FIMIN(i,j,k,l,m)
 int i,j,k,l,m;
 { float df,dfi;
   df=conf(i,k,m)-conf(j,l,m);
```

```
130     dfi=(ABS(df)<ABS(df-2*PI))? ABS(df):ABS(df-2*PI);
131     return (ABS(df+2*PI)<dfi)? ABS(df+2*PI):dfi;
132   }
132   }
```

Комментарии. Только в целях сокращения объема программы нами не выделены вполне естественные процедуры, реализующие пункты описанного алгоритма. Надеемся, что некоторые комментарии принесут вам пользу в вашей дальнейшей практике. При этом мы будем ссылаться на порядковые номера строчек в программе (выше вы увидели *нечто* в стиле Си-Бейсик).

3. Эта строка иллюстрирует любопытную вещь: переменные x и y обмениваются значениями так, что для этого не требуется *дополнительная* переменная (мы сэкономили память!). Однако надо иметь в виду, что этот способ пригоден с некоторыми оговорками, касающимися точности представления переменных x и y. Так, например, если $x=1E20$, а $y=1E-20$, то такой обмен будет некорректным. В нашем случае макроопределение *SWAP* используется для целых переменных небольшого порядка.

4. Макроопределение *NORM* приводит угол к углу в интервале $[0,2\pi[$.

7—8. Макроопределения для более удобного и естественного обращения к многомерным, динамически распределяемым массивам.

9. Вектор *cv*, сопоставляемый массиву *conf*, предназначен для хранения конфигураций робота.

12. Исходные данные в порядке, описанном в задаче.

13. x и y — координаты вводимой точки;

fl1 — «флаг» для прекращения дальнейшего перебора деталей в цепочке;

n — число *достижимых* точек;

or $[0, n]$ — множество конфигураций робота;

posl $[0, n]$ — последовательность деталей, дающая минимальное суммарное время;

num $[0, n]$ — порядковые номера точек в порядке их ввода;

seg $[0, n]$ — очередная сгенерированная последовательность обхода точек;

fl — порядковый номер точки с координатами $(0, 0)$.

14. t — время обработки последовательности деталей; *T_min* — минимальное время обработки всех n точек; *alfa* и *beta* — углы из формул (4) и (5);

dt [2] — времена поворотов звеньев в первом и втором шарнирах;

tv — вектор, сопоставляемый массиву *tim*, предназначен для хранения времен перехода от одной точки к другой (t_{ij}^{kl});

FIMIN — функция, дающая минимальное по абсолютной величине значение угла поворота в диапазоне $[0, \pi[$.

15—42. Ввод данных с контролем. Если ввод данных осуществлялся бы из файла, то в ответ на неверно введенное число программа должна прекратить работу.

43—46. Выделение места под массив номеров деталей и под массив конфигураций робота только для начальной (нулевой) точки и запись начальных углов.

47—63. Ввод координат точек; определение достижимости точек; выделение дополнительного места для записи конфигурации робота и номера детали; запись номера доступной детали; вычисление и запись углов в массив конфигураций; присваивание «флагу» *fl* номера точки, если ее координаты $(0, 0)$. Обратите внимание на то, что в строке 50 проверяется выполнение не условий (2) и (3), а эквивалентных (с точки зрения математики) условий:

$$(L1+L2)^2 \leqslant r^2, \qquad (13)$$

$$(L1-L2)^2 \geqslant r^2. \qquad (14)$$

Однако нет никакой идентичности этих условий неравенствам (2) и (3) с точки зрения *вычислительной* математики. Так, если $L1=1$ и $L2=4$, а $x=3$ и $y=4$, то проверка (13) показывает, что точка (x, y) лежит строго на границе области достижимости:

$$(1+4)^2=3^2+4^2,$$

в то время как нет никакой гарантии, что вычисление r с использованием арифметики с плавающей точкой не приведет к результату $r=5.000001$, который означает, что точка недоступна. Задаваться при этом любой, сколь угодно малой погрешностью совершенно неправомерно, так как в условии задачи оговорено, что все исходные данные — целые числа.

64—67. Выделение необходимого места в памяти для размещения массивов *tv*, *seg*, *posl* и *or*.

68—85. Заполнение массива времен перехода t_{ij}^{kl}. Особым образом обрабатывается случай перехода в точку $(0, 0)$. В строках 74 и 75 все четыре времени являются временами перехода *из* некоторой точки *в* точку $(0, 0)$. Симметричные элементы матрицы не записываются, поскольку в них нет необходимости. В строке 82 записываются сразу два (симметричных) элемента матрицы T.

86—87. Установка начальной последовательности деталей A_1.

88—118. Подсчет минимального времени *T_min* и максимального числа деталей *N_max*.

92—93. Вычисление дополнительного времени для разворота первого звена при переходе в i-ю точку из точки (0, 0).

94—95. Вычисление времени перехода из точки (0, 0) в i-ю точку.

97. Подсчет времени перехода в общем случае.

100. В случае, если $t>T\_fix$ (т. е. уже не нужен подсчет числа деталей для пункта в) задачи) и *T_min*, полученное ранее, меньше t, то «флаг» *fl*1 устанавливается в положение 0 и дальнейшее выполнение цикла 90—101 прекращается.

102—104. В случае полного прохода цикла при условии, что $t<T\_min$, записывается новое значение *T_min* (103) и запоминается последовательность обхода точек (103—104) в массиве *pos*1.

105—108. Для внутреннего цикла (89—118) формируется новое множество конфигураций. При этом вектор *or* рассматривается как двоичное число и операторы 105—108 выполняют функцию прибавления единицы в $(i-1)$-м разряде, поскольку именно при значении $(t-1)$ был прекращен просмотр предыдущей цепочки. Единица прибавляется к наименьшему разряду, большему или равному $(i-1)$ и содержащему 0. Значение этого разряда устанавливается равным 1, если это не самый старший разряд, и равным 0 — в противном случае. Это же значение (0 или 1) принимает переменная *kod*1, управляющая выполнением внутреннего цикла. Значение $kod1=0$ приводит к прекращению выполнения внутреннего цикла (89—118). Все последующие разряды «двоичного числа» *or* устанавливаются равными нулю (107—108). После выхода из внутреннего цикла (по условию $kod1=0$) необходимо сгенерировать новую последовательность обхода. Эту функцию выполняют операторы 110—117.

Остановимся подробнее на этом фрагменте программы. Казалось бы, самый естественный и простой способ генерации перестановок — использование рекурсивной процедуры, описанной в каждом учебнике по программированию. Однако следует принять во внимание, что при большом значении n очень быстро переполняется стек вызова подпрограмм со стандартным завершением «*Stack overflow*». Чтобы избежать этого, мы использовали более медленный алгоритм попарных перестановок чисел в последовательности *seg*. При попытке сгенерировать последовательность, следующую за $\{n, (n-1), ..., 2,1\}$, цикл 110 завершается при $i=1$ и переменная *kod* становится равной 0 (117). На этом завершается выполнение внешнего цикла.

119—123. Печать результатов счета.

Ошибки. Сначала о «тактических» просчетах. Некоторые участники олимпиады построили решение задачи не на полном переборе всех вариантов обхода, а на использовании некоторых эвристических правил, например «сканировании» звеньями по 1°.

Многие участники не увидели, что могут быть и недостижимые точки, а те, которые это заметили, не всегда правильно использовали неравенства для проверки ((13) и (14) вместо (2) и (3)).

Отнюдь не во всех программах звенья робота при переходе из точки в точку описывали углы, меньшие π по абсолютной величине. Далеко не все участники заметили, что в общем случае возможны две конфигурации робота при обработке детали.

К сожалению, очень многие (и часто хорошие) программы не содержали контроля входных данных и случаев деления на 0.

Не все участники, реализовавшие полный перебор с использованием рекурсивных процедур, избегали повторных обработок некоторых деталей в различных конфигурациях.

И уж совсем никто не предусмотрел сложного перехода через точку (0, 0) (хотя мы этого и не очень-то ожидали).

Здесь мы не касаемся вопросов оформления программ, использования метода ветвей и границ и элементарных ошибок в тригонометрии.

Литература

1. *Вентцель Е. С.* Исследование операций. М.: Наука, 1988.

Новые программные продукты НТВМП «Логосъ»

Новая версия сервисного расширения Сириус — незаменимой интерактивной оболочки языка BASIC. Работает на ПЭВМ «Агат-7» и «Агат-9». Тот, кто хоть раз пользовался нашей системой Сириус, ни за что уже не захочет работать на стандартном BASIC.

Стоимость 2 дискет с расширением Сириус — 350 рублей.

Инструментальная среда КОМПОЗИТОР — работает на ПЭВМ «Электроника МС 0202» (УКНЦ), «Агат-7», «Агат-9»,— лучший из известных нам музыкальных редакторов для данных типов ПЭВМ. Только используя КОМПОЗИТОР, вы сможете писать музыкальное сопровождение к любым вашим программам.

Стоимость 2 дискет со средой КОМПОЗИТОР — 450 рублей.

Авторская среда подготовки уроков МИНОС — работает на ПЭВМ «Электроника МС 0202» (УКНЦ), «Агат-7», «Агат-9»,— исключительно удобное, доступное учителю-предметнику, средство, позволяющее ему готовить и проводить в классе ПЭВМ свои уроки.

Стоимость 10 дискет с МИНОС и наполнением по информатике, математике, физике — 2150 рублей.

Спешите оформить заказ по адресу: 428034, г. Чебоксары, а/я 99, НТВМП «Логосъ», тел. 42-82-77, 22-82-18. Р/с 609201 в Коммерческом банке «Чувашкредитпромбанк» г. Чебоксары. МФО 286352.

Ленинградский производственный кооператив «Компьютерные игры» предлагает

программное обеспечение для компьютеров БК-0010, 0010.01, ДВК-2М, 3М, КУВТ-86, УКНЦ, Синклер, IBM PC. Кооператив располагает обширным банком игровых, учебных, системных, прикладных программ.

Расценки значительно ниже государственных.

Списки программ, при указании типа ЭВМ, высылаются бесплатно.

Для пользователей БК-0010, БК-0010.01:

пакеты новых игровых, системных и прикладных программ;

бесплатная запись всем заказчикам турбокопировщика HELP7M, позволяющего на каждой кассете пользователя размещать в три раза больше программ при многократном увеличении надежности.

Переписка: 189510, Ленинград — Ломоносов, а/я 649.

Ленинградский производственный кооператив «ОСТ» предлагает

программное обеспечение для компьютеров БК-0010, 0010.01, ДВК-2М, 3М, КУВТ-86, УКНЦ и для всех совместимых с IBM PC.

Кооператив располагает обширным банком игровых, учебных, системных, прикладных программ.

Расценки значительно ниже государственных.

Списки программ (при указании типа ЭВМ) высылаются бесплатно.

Наш адрес: 199034, Ленинград, а/я 437.

КЛИП

Крутые игры — 3

Вот и пришло время заканчивать остросюжетный цикл «Супермен». Все решит последняя схватка. Брюс Ли добрался до мрачного бункера Во Ланда, и вам пора ввести в ПМК программу Р. Шубочкина (Мурманск).

	0	1	2	3	4	5	6	7	8	9
0	В/О	ИП6	КППВ	КЗН	ИП0	+	П0	ИП1	КППВ	Кх<0В
1	1	П5	2	КПП7	Кх=09	КИП3	КИП3	КВП9	1	/-/
2	П0	F10ˣ	8	+	КИНВ	П6	7	/-/	П2	В/О
3	Кх=0Е	2	П5	3	КПП7	Кх=09	КИП3	КВП9	3	П5
4	4	КПП7	Кх=09	КИП3	КИП3	ИП6	КСЧ	6	X	К[x]
5	В↑	КСЧ	2	X	К[x]	ИП2	+	П2	F10ˣ	X
6	8	+	ИП6	К0	П6	ИП1	F√	ИП3	Fx≥0	95
7	КВПС	ИП0	F10ˣ	X	8	+	КИНВ	П6	ИП0	ИП2
8	−	Кх≥0Д	2	ИП0	ИП2	−	−	Кх=0А	КИП1	КИП1
9	КИП1	FBx	ИП5	−	В/О	ИП4	С/П	В↑	В↑	В↑
A	В↑	В↑	В↑	ИПС	Fcos					

Исходные данные программы «Карате»:

```
71 П7 30 П8 45 П9 91 ПА 97 ПВ 100 ПС 18 ПД 38 ПЕ
```

Сигнал победы «ВСЕ»:

```
831 КИНВ П4 (ВСЕ)
```

В П1 и П3 заносится количество энергии (например, 10) вашей и противника соответственно. Приготовились? Начинаем! БП 18 С/П.

Вы настигли Во Ланда в последней, самой уединенной комнате. Вначале высвечивается картина поединка. Комната разделена на 7 позиций; «8» слева — это вы, Во Ланда пока не видно, он за правым краем экрана. Сделав шаг вперед, вы окажетесь на второй позиции, теперь «8» обозначает дверь, через которую вы вошли. Обычная стойка, в которой вы находитесь, обозначается буквой Е. Сейчас можно с помощью переключателя угловых мер выбирать дальнейшие действия: Р — шаг назад, ГРД — остановка, Г — шаг вперед (на одну позицию). Не затягивайте с переключением, иначе ПМК не отреагирует на ваш приказ.

Следующий видеоблок показывает остаток вашей энергии, уменьшившейся на величину приложенных усилий и в результате нападений Во Ланда. Если энергия кончится, вы погибли — ЕГГОГ.

Не пора ли нанести удар? Установите переключатель: Р — удар ногой (на экране обозначается L, действует на расстоянии трех позиций), ГРД — удар рукой с подсечкой (обозначается С, действует на расстоянии двух позиций), Г — удар рукой (Г действует только на соседнюю позицию). Удар выполняется из положения, которое вы приняли после перемещения в предыдущем блоке!

Противник не стоит на месте. Он бьет на две позиции (удар, обозначаемый С), но опаснее всего его неузнаваемость. Этот оборотень может прикинуться на экране любым символом и даже стать невидимым. Тогда придется воевать вслепую. Если вы окажетесь на одной позиции с Во Ландом, ПМК разнимет вас и поместит в крайние позиции комнаты.

Вашу победу засвидетельствует слово ВСЕ на экране. Попробуйте победить еще раз, снизив запас своей

энергии.

Итак, Во Ланд окончательно вышел из игры и не представляет больше никакой опасности. Брюс Ли встречается с Марикой, а мы не станем подглядывать и займемся делами КЛИПа.

Для новичков напомним, что членом КЛИПа (Клуба любителей игровых программ для ПМК) может стать каждый независимо от возраста и социального положения, знания иностранных языков и восточных единоборств. Берите лист бумаги и пишите Ф. И. О., возраст, профессию, адрес, телефон, тип ПМК, какие программы (в том числе собственные) и литературу имеете, какие типы игр (головоломки, экономические, лабиринтовые, динамические и т. д.) вас интересуют. Вложите лист в конверт вместе с другим конвертом и посылайте в редакцию «ИНФО». Вы будете внесены в картотеку клуба и сможете найти друзей-единомышленников в любом уголке нашей страны.

Главная проблема общения — язык. Выработка языка записи программ, удобного для публикаций и обмена программами, является насущнейшей задачей КЛИПа. Программы цикла «Супермен» печатались на традиционном, «журнальном» языке, достаточно наглядном, но не удобном тем, что в нем используются буквы как латиницы, так и кириллицы. На пишущей машинке такую программу целиком не напечатаешь, приходится вписывать от руки. Это послужило причиной появления различных языков, диалектов и даже сленгов, что затрудняет обмен программами.

Руководитель КЛИПа М. Храмов создал язык УНИКАЛ-2. Недостаток последнего очевиден (малая наглядность), достоинство (использование только кириллицы) также налицо.

МК 61 (МК 52)	УНИКАЛ-2	МК 61 (МК 52)	УНИКАЛ-2	МК 61 (МК 52)	УНИКАЛ-2	МК 61 (МК 52)	УНИКАЛ-2
.	ТЧ	tg	ТН	max	МХ	L1	Ц1
/-/	СЗ	π	ПИ	A	КН	x<0	МН
Cx	СХ	↓	КР	V	ДЗ	L0	Ц0
Bt	ВТ	x^2	КВ	⊕	СД	x=0	РВ
Bx	ВХ	1/x	ОБ	инв	ИН	П↔x	ПХ
:	/	x^y	ХУ	Bx	ВУ	a	A
$\overleftarrow{XY}$	ВВ	K-	ЕГ	At	АД	b	B
10^x	ДХ	(вызов	ЕГГОГ)	↑↓	РП	c	C
e^x	ЕХ	◊	ВН	x↔П	ХП	d	Д
lg	ЛГ	⟶	ГД	С/П	СП	e	E
ln	ЛН	⟵	ГМ	В/О	ВО	⊞Р	ШВ
$\sin^{-1}$	АС	⟶	ЧД	НОП	НО	⊞Г	ШН
$\cos^{-1}$	АК	⟵	ЧМ	x≠0	НР	ABT	АВ
tg^{-1}	АТ	!x!	МД	L2	Ц2	ПРГ	ПР
sin	СН	[x]	ЦЛ	x≥0	БР	CF	С⊕
cos	КС	{x}	ДР	L3	Ц3	PZ	Р⊕

Остальное без изменений

«Противоположно-параллельный» подход — записывать программу только цифрами, воспроизводя коды ПМК. Такой вариант по сравнению с УНИКАЛОМ-2 проигрывает в отношении мнемоничности.

Так что же выбрать? «Журнальный», «кодовый», УНИКАЛ-2? А может быть, вы предложите свои варианты языка? В любом случае решать, на каком языке записывать программы на страничке КЛИПа, будут читатели. Пишите нам!

Ждем также ваших оценок (по 10-балльной системе) опубликованных программ по следующим параметрам: сервис, игра, общая оценка. А главное, ждем ваших программ, идей, нестандартных приемов, новых увлекательных сюжетов, предложений по ведению рубрики. Если что-то не получается — не отчаивайтесь! КЛИП вам поможет!

В. МОСКОВЦЕВ

Ассемблер для ПМК

А вот подход к проблеме языка для записи программ ПМК другого автора.

Язык клавиатурных символов, повсеместно используемый для записи программ ПМК, имеет два существенных недостатка:

трудно понять математический смысл записанной на нем программы;

при составлении программы приходится с самого начала распределять память, и малейшее изменение (а их обычно приходится делать) в распределении вынуждает тщательно перепроверять весь текст.

Естественно возникает мысль об использовании при программировании на ПМК некоего подобия ассемблера. Попробуем разработать его.

Прежде всего введем упрощенные обозначения для команд. Вместо ИП3, П3, КИП3, КП3, 3 будем писать 3, 3, '3, '3, (3). Изменим также команды переходов (табл. 1) и еще несколько символов (табл. 2). Остальным командам дадим новые обозначения по вполне очевидной аналогии или оставим старые.

Таблица 1

Старое обозначение	БП 00	К БП 0	F ПП 00	К ПП 00	F x<0 03	K x<0 3	F L1 03
Новое обозначение	[00]	'[0]	[П00]	'[П0]	[<03]	'[<3]	[103]

Таблица 2

Старое обозначение	В/О	КНОП	С/П	F Bx	XY	F↻	Cx	F, arccos
Новое обозначение	[П]	[]	[.]	:	--	0	()	arccos

Попробуйте теперь переписать несколько программ из своего архива новыми обозначениями. Выгоднее использовать короткие строки, не обязательно одинаковой длины. Адрес каждой команды указывать не стоит, достаточно адреса первой команды строки, да и то не для всех строк. И вот — более ясным становится смысл программ, вдоль текста остается широкое поле для размещения инструкции, сократился объем ненужной (а следовательно, вредной) информации.

Найдем, к примеру, значение выражения $(ax+by)/cz$. Исходные значения a, x, b, y, c, z запишем в регистры 1, 2, 3, 4, 5, 6, а выполнение программы будем начинать с адреса 00.

Программа	Инструкция
00: 1 2 × 3 4 × +	1. Ввод по таблице:
07: 5 6 × ÷ [·] {2} (число в фигурных скобках указывает на параграф инструкции)	Переменная: a x b y c z Регистр: 1 2 3 4 5 6 Значение: Счет с адреса 00 2. Результат в регистре X

А теперь перепишем программу так.

L: a x × b y × +
c z × ÷ [·] {2}

Инструкция остается такой же, только добавляется табличка
Метка: L
Адрес: 00

Можно ли такой программой пользоваться для работы? Вполне. Ее, правда, несколько труднее вводить, нужно смотреть и на текст, и на таблицу перевода, но зато значительно легче составлять, легче понять смысл. От распределения памяти такая запись не зависит, любые изменения коснутся только табличек. При решении сложных задач, требующих полного использования ресурсов ПМК, отделить процесс составления программы от распределения памяти невозможно, но все равно приведенные выше новые обозначения команд выгоднее, чем обычная запись на языке клавиатурных символов.

Приведенный пример демонстрирует только идею и не заменяет строгого описания языка, нужного для обучения началам информатики с использованием ПМК, однако для подготовленного человека достаточно идеи. Шестилетний опыт автора показал эффективность такой записи; попробуйте использовать ее и делайте выводы сами.

А. АБЕЛЬСКИЙ

Экологический центр на базе школы

Проблемы экологии уже давно волнуют человеческую мысль, но только в последнее десятилетие защита окружающей среды вышла на первый план в сознании политиков, ученых, всех обеспокоенных за будущее людей, наряду с проблемами ограничения вооружений.

С прошлого учебного года группа учителей 49-й московской школы, специалистов по информатике, географии и биологии, решила привлечь своих школьников к занятиям по экологии с использованием компьютеров, что помогло бы сделать предмет живым и привлекательным для детей. Так родилась идея написания экологических моделирующих компьютерных программ (ЭМКП). Старшие школьники, овладев необходимыми знаниями в программировании, научились навыкам написания ЭМКП.

Так на базе школы было организовано детское эко-

логическое агентство (ДЭА) «Мониторинг», зарегистрированное как малое предприятие, специализирующееся на создании ЭМКП. На городских соревнованиях компьютерных программ в прошлом году работы членов ДЭА заняли первое место. У ДЭА уже есть опыт продажи детских программ. По этим программам учатся их младшие друзья в нашей школе.

В активе «Мониторинга», не только написание программ, но и переписка с учащимися Англии и США, чему способствует углубленное изучение английского языка в школе. В процессе обмена письмами мы заинтересовались проектом «Waterways», который предполагает исследование реки учащимися средней школы. Кроме того, представители Лондонского университета, посетившие школу, предложили обмен информацией по различным аспектам охраны природы, таким, как радиоактивность, кислотные дожди и наблюдения за животными. Мы надеемся, что данная работа будет проводиться и другими школами в разных регионах.

С этого года начинается эксперимент по преподаванию в V классе ряда разделов курса географии на компьютерах. Для этого будут использоваться разработки Национального общества США — программы «Hello» и «Acid Rains». Однако необходимо учитывать разницу между советской и американской методиками преподавания предмета.

Все перечисленные задачи требуют создания банка данных и накопления в нем разнообразной экологической информации. В настоящее время мы разрабатываем программу, позволяющую сохранять результаты разнообразных измерений и в дальнейшем работать с ними. Собранный материал планируется использовать самыми разными способами: включать в создаваемые ЭМКП, обмениваться с другими школами, использовать непосредственно на уроках.

Но основная деятельность ДЭА «Мониторинг» — это создание программного продукта — экологических моделирующих программ.

До приобретения IBM-совместимой техники учащиеся постигали азы программирования на БК. Ими были созданы два пакета программ в рамках экологической тематики. Первый пакет программ ориентирован на старшеклассников, главной его темой являются экологические катастрофы, и называется он «ЭКО-ЧП».

В «ЭКО-ЧП» входят четыре программы, рассматривающие разные источники экологических бед:

1. «Нефтяное пятно» — «SPOT1» + «SPOT2» + + «COAST».
2. «Радиоактивная свалка» — «RADIOACTIVE ZONE».
3. «Лесной пожар» — «FOREST».
4. «Загрязнение реки» — «RIVER».

Все программы организованы по разным принципам, и поэтому универсального рецепта работы с ними не имеют.

Второй пакет предназначен младшим школьникам и затрагивает вопросы природопользования в сельском хозяйстве.

Мы надеемся, что эта тема заинтересует учащихся сельских школ, но, возможно, она окажется полезной и городским школьникам. В компьютерное обеспечение входят четыре программы:

1. «Деревня» — «VILLAGE».
2. «Обработка поля» — «FIFLD».
3. «Инкубатор» — «CHICKEN».
4. «Рыбак» — «FISHER».

Включение школы в проект «Пилотные школы» открыло перед учащимися возможность работать на IBM. Приобретенные знания в программировании, экологии, экономике и работа на мощной технике позволили создать более интересные разработки.

Одна из них — программа «RIVER_Z» (авторы — ученики XI класса М. Козловский, Э. Литвак, Г. Трофимов), прототипом которой является «RIVER».

Программа позволяет смоделировать муниципальный контроль за деятельностью предприятий, ведущих сброс сточных промышленных вод в реку. Идея программы очень проста. В нашем распоряжении имеется плавающий по реке корабль. Клавишами «↓», «↑», «→», «←» можно смещать индикатор соответственно к левому или правому берегу, двигать его вверх по течению, ускорять или замедлять движение до полной остановки. Касаться берегов индикатором нельзя, так как при этом он садится на мель и теряет свободу перемещения.

В начале работы с программой можно использовать досье, где указаны все 9 предприятий, расположенных по берегам реки.

Итак, корабль плывет по реке и дает цветовую и звуковую сигнализацию, попадая в загрязненную воду.

Вернувшись в город, можно зайти в суд и подать иск на предприятие, виновное в загрязнении. Если предприятие определено неверно, то все убытки возмещаются из ваших средств. Также можно зайти в мэрию, где вы, возможно, получите инвестиции и финансирование на проведение природоохранных мероприятий, сможете наложить санкции на предприятия-нарушители.

Для оценки экологического состояния реки играющему предоставлена возможность просмотра некоторых наиболее важных показателей в любом месте реки. К таковым отнесены ПДК вредных веществ, разделенные на 8 групп: температура, мутность, минерализация воды, состояние биосферы и т. д. (по показателям количества рыбы, микроорганизмов и прочих составляющих флоры и фауны).

Имея необходимое количество денег, играющий приобретает в магазине стационарные буи-лаборатории, которые можно расставить в любом месте реки (клавиша *F3*), причем они невидимы, так как должны проводить наблюдения скрыто от предприятий и хозяйств. Увидеть их можно с помощью специальной клавиши (*F2*). Если буй окажется рядом с индикатором, он будет «снят» и перенесен на новое место.

Играющий может вернуться в город в свой рабочий кабинет, где на плане реки изображены предприятия, хозяйства и расставленные буи. Перемещая указатель от буйка к буйку, играющий может просмотреть на мониторе столбчатые диаграммы, отражающие состояние реки. Если буй находится у предприятия или хозяйства, пользующегося речной водой, то на диаграммах отражается его воздействие на экологию реки, в случае произведения залпового сброса вредных веществ в воду на диаграммах будет показано значительное отклонение от норм пропорционально времени действия.

По каждой диаграмме пользователь может вызвать окно сообщений, в котором поясняются все показатели. В досье о предприятиях и хозяйствах, организованном как микробаза данных, кроме экономико-социальных показателей содержится информация о характере загрязнения и составе сбросов.

В кабинете имеется телефон, по которому поступают сведения, позволяющие играющему предугадать возможного загрязнителя. Загрязнение реки происходит согласно следующему сценарию.

Крупная ферма не оборудована очистными сооружениями и поэтому производит постоянный сброс в реку воды с повышенным содержанием загрязняющих веществ. НИИхиммаш, химзавод, ТЭЦ, нефтехранилище и автобаза производят залповые сбросы с различной вероятностью. После дождей происходит смыв загрязняющих веществ с полей, территории свалки. Обильные канализационные стоки также повышают уровень загрязнения реки.

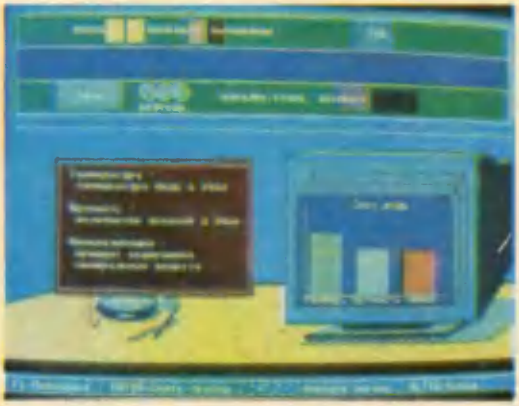
В ДЭА царит атмосфера научного творческого коллектива. Программы создаются целой группой школьников. Работая над своей частью программы, каждый учащийся всегда может использовать находки своих товарищей. Так, для создания многокрасочных рисунков ученик X класса. Г. Трофимов написал специальную программу — графический редактор «*SPRITE*», позволяющую рисовать картинки, делать мультипликацию. Все рисунки в «*RIVER_Z*» созданы именно с помощью этой оригинальной программы.

В настоящий момент агентство располагает целым набором ЭМП на БК и *IBM* с методическими разработками. Формы работы с ними можно предложить следующие:

отдельные уроки по биологии и географии в кабинете информатики, посвященные экологической тематике;

примеры моделирующих (и воспитывающих) программ для урока информатики;

факультатив по экологии.



Планы ДЭА «Мониторинг» обширны. Это и создание целого курса по экологии для возможного использования в школах страны, расширение связей со школами в СССР и за рубежом. Пополнение экологического банка данных по Москве, обмен информацией с заинтересованными лицами.

А. САХАРОВ, учитель географии
Л. КЛЯЙМАН, учитель информатики
Г. ТРОФИМОВ, ученик XI класса
М. КОЗЛОВСКИЙ, студент

Подведем итоги

Публикуемые ниже материалы подготовлены Л. Е. Самовольновой, координатором целевой комплексной программы «Информатизация общего среднего образования», разработанной рабочей группой секции школ Общесоюзного научно-методического совета Гособразования СССР в 1990 г.

Программа была одобрена Коллегией Гособразования СССР в декабре 1990 г. На финансирование работ в 1991 г. было выделено 520 тыс. рублей, однако 100 тыс. рублей из них так и не дошли до творческих коллективов трех институтов АН СССР в связи с происходившими в стране событиями.

К моменту, когда читатель получит этот номер журнала, результаты программы будут систематизированы, изданы (надеемся) и распространены по общеобразовательным школам и институтам усовершенствования учителей и повышения квалификации педагогических кадров.

Разработаны и переданы в издательство «Просвещение» новые программы по курсу «Информатика» для классной и внеклассной работы с учащимися различных возрастных групп. По плану издательства этот сборник должен попасть в школы до 1 сентября 1992 г.

Перечень программ по курсу информатики

1. Программа по курсу информатики для VIII—IX классов средней школы. Авторы: *А. Г. Кушниренко* и др., Москва.
2. Программа по курсу информатики для VIII—IX классов средней школы. Авторы: *А. Г. Гейн* и др., г. Екатеринбург.
3. Программа по курсу информатики для VIII—IX классов средней школы. Авторы: *А. А. Кузнецов* и др., Москва.
4. Программа по курсу информатики для VIII—IX классов средней школы. Авторы: *Б. Г. Киселев* и др., Москва.
5. Программа по курсу «Роботландия» для начальной школы. Авторы: *Ю. А. Первин* и др., г. Переславль-Залесский.
6. Программа факультативных занятий по курсу информатики. Автор *В. П. Федотов*, г. Иваново.
7. Программа факультативного курса «Основы программирования» для IX(X) класса средней школы. Автор *Г. К. Григас*, г. Вильнюс.
8. Программа для школ (классов) с углубленным изучением курса «Основы информатики и вычислительной техники» для VIII—XI классов. Авторы: *А. Г. Гейн, А. И. Сенокосов*, г. Екатеринбург.

Несомненным вкладом в развитие школьной информатики послужат программно-методические комплексы по информатике и другим школьным дисциплинам для учащихся различных возрастов и различной подготовки, созданные в рамках программы известными коллективами разработчиков под руководством ученых: Л. В. Городней (Новосибирск), В. Ф. Шолоховича (Екатеринбург), А. Г. Гейна (Екатеринбург), Г. К. Григаса (Вильнюс), А. Г. Кушниренко (Москва), И. В. Роберт (Москва), Ю. А. Первина (Переславль-Залесский) и др.

Список пакетов ПС учебного назначения, разработанных в рамках целевой комплексной программы «Информатизации общего среднего образования» в 1991 г.

Все пакеты ПС учебного назначения приняты Всесоюзной экспертной комиссией среднего уровня образования и рекомендо-

ваны для распространения в массовой школе, институтах усовершенствования СССР и педагогических вузах.

1. Пакет «Торт» — набор программ, которые используются как инструментарий для работы младших школьников с текстовой информацией. В состав входят тренажеры для работы с клавиатурой и исправления ошибок (Курсор, Правилка, Привет) и адаптированный текстовый редактор Микрон, который наряду с формированием навыков работы с текстами эффективно используется для диктантов и упражнений по родному языку. (IBM PS/2, 3, 5" малое предприятие «Роботландия», руководитель — д. п. н. Первин Ю. А.).

2. ППС «Экологические основы природопользования». Педагогическое программное средство предназначено для использования в преподавании факультативного курса по основам экологии, использования природных ресурсов для учащихся X—XI классов средних школ. (IBM PC, 3,5", Свердловский ГПИ, руководитель — Шолохович В. Ф.).

3. ППС «Учебный исполнитель алгоритмов Паркетчик». ППС предназначено для использования в преподавании курса ОИВТ в V—VII классах средних школ. (КУВТ «Корвет», 5,25", Свердловский ГПИ, руководитель — Гейн А. Г.).

4. Система КуМир. Система КуМир с исполнителями поддерживает школьный курс информатики и согласована по структуре, терминологии и методологии со школьным учебником информатики (А. Г. Кушниренко, Г. В. Лебедев, Р. А. Сворень — Основы информатики и вычислительной техники: Пробный учебник для средних учебных заведений. М.: Просвещение, 1990). Она предназначена как для работы на уроках информатики, так и для факультативных занятий и кружков на ПЭВМ «Ямаха MSX-1, MSX-2 и "Корвет", ("Ямаха", "Корвет", 3,5". МГУ им. М. В. Ломоносова, руководитель — к. ф.-м. н. Кушниренко А. Г.).

5. Банк данных «Мир профессий» создается для решения задач профориентации школьников, повышения эффективности труда преподавателей, библиотекарей школьных и детских библиотек, сотрудников региональных профориентационных центров. Банк данных представляет собой каталог, содержащий в первой редакции базу данных по профессиям сферы обслуживания, и включает: данные о профессиях, о сферах деятельности (отраслях хозяйства), библиографические описания книг и статей о профессиях с аннотациями, статьи психологов и других специалистов в выборе профессии. IBM PC/XT/AT, 5,25". Библиотека им. В. И. Ленина, руководитель — Лаврентова О. А.).

6. Пакет баз данных для IBM-совместимых ПЭВМ «Делопроизводство в средних учебных заведениях». Пакет предназначен для автоматизации ведения делопроизводства во всех типах средних учебных заведений и органах управления народным образованием. В 1991 г. пакет получил серебряную модель ВДНХ СССР. (IBM, 5,25". НИИ содержания образования и учебной книги АПН СССР, руководитель — к. п. н. Роберт И. В.).

7. Обучающая программно-методическая система «Многогранники». Пакет предназначен для использования в процессе изучения темы «Многогранники» курса «Стереометрия», ориентирован на формирование основных стереометрических понятий, умений построения сечений призмы плоскостью, развития компонентов наглядно-образного и теоретического типов мышления ("Ямаха", 3,5". НИИ содержания образования и учебной книги АПН СССР, руководитель — к. п. н. Роберт И. В.).

8. Пакет «Исследование функций». Пакет предназначен для обучения исследованию функций по алгоритмическим предписаниям; имеется возможность обращения к теоретическому материалу, использования калькулятора. Связи между кадрами определяются в зависимости от уровня правильности полученных от пользователя ответов на предложенные вопросы и вызовами, задаваемыми самим пользователем ("Ямаха", 3,5". НИИ содержания образования и учебной книги АПН СССР, руководитель — к. п. н. Роберт И. В.).

9. Русский язык. Правописание мягкого знака после шипящих. Программа предназначена для отработки орфографических навыков на этапе закрепления и проверки их сформированности. Позволяет получать полную информацию о сделанных учеником ошибках и организовать работу над ошибками с помощью генерации упражнений ("Ямаха", 3,5", НИИ СО и УК АПН СССР, руководитель — д. п. н. Зельманова Л. М.).

10. Система имитационного моделирования «Опыт Милликена». МП система предназначена для воспроизведения одного из фундаментальных физических опытов по определению заряда электрона. Реализует экспериментально-исследовательскую деятельность учащихся. Позволяет в ходе изменения параметров и условия эксперимента выявить закономерность изменения величин электрических зарядов, определить

значение минимального заряда ("Ямаха", 3,5", НИИ СОиУК АПН СССР, руководитель — Разумовская Н. В.).

11. Пакет «Перспектива-1». Пакет предназначен для поддержки школьного курса информатики. Содержит набор тренажерных, демонстрационных, инструментальных и контролирующих ППС, обеспечивающих учебную деятельность школьников по всем разделам курса. Пакет сопровожден методическими рекомендациями для учителей по его применению в учебном процессе ("Ямаха", 3,5", НИИ общего среднего образования АПН СССР, руководитель д. п. н. Кузнецов А. А.).

12. Пакет «Перспектива-2». Пакет дополняет и развивает программное обеспечение курса ОИВТ, представляет школьникам возможность приобретения информационных навыков и умений на различном предметном материале, что способствует развитию их интереса, мотивации к учению ("Ямаха", 3,5", НИИ общего среднего образования, руководитель — д. п. н. Кузнецов А. А.).

13. Редактор контролирующих программ КТОР. Редактор позволяет непрофессиональному пользователю разрабатывать тренажерные и контролирующие программы по любому предмету, архивировать и систематизировать данные по результатам обучения школьников, вести «электронный» журнал ("Ямаха", 3,5", НИИ общего среднего образования, руководитель — д. п. н. Кузнецов А. А.).

14. Редактор дорожных ситуаций. Редактор предназначен для обучения правилам уличного движения учащихся школ, ПТУ, слушателей курсов ДОСААФ, для профессиональной подготовки водителей. Позволяет непрофессиональному пользователю создавать из готовых графических фрагментов различные дорожные ситуации, варьируя значениями светофора, набором дорожных знаков, сигналами регулировщика и т. д. К созданной «картинке» можно подготовить перечень вопросов, комментариев и т. п. Редактор дает возможность оценивать ответы обучаемых, систематизировать и архивировать их ("Ямаха", 3,5", НИИ общего среднего образования АПН СССР, руководитель — д. п. н. Кузнецов А. А.).

15. Пакет «Begek арифметика». Пакет содержит комплект игровых программ, предназначен для поддержки обучения математике в I классе. Может быть использован для подготовки детей в школе ("Ямаха", 3,5", НИИ общего среднего образования АПН СССР, руководитель д. п. н. Кузнецов А. А.).

16. «Пифагор». Программный комплекс для обучения программированию на основе графического языка образов П-схем (IBM PC/XT/AT PS/2, EGA, VGA, 640Кб, 5,25", 1.2 Мб или 3,5", 1,44 Мб, Институт математики и механики Ур. отд. АН СССР, руководитель — к. ф.-м. н. Прохоров В. В.).

17. Учебное пособие «Задачи по алгоритмизации и программированию». Программный комплекс предназначен для обучения решению задач по теме «Алгоритмы», языку Турбо, Паскаль Версия 6 (IBM PC, 5,25", Институт математики и информатики АН Литвы, руководитель — к. ф.-м. н. Григас Г. К.).

18. ПМК по информатике для учащихся III—VIII классов. ПМК ориентирован на самообучение и на работу с учащимися III—VIII классов. ПМК включает учебно-игровой редактор-имитатор для конструирования игр и обучающих программ, варианты исполнителей для языка Робик на базе системы программирования Рапира, учебную базу данных ("Ямаха", 3,5", Институт системы информатики Сиб. отд. АН СССР, руководитель — к. ф.-м. н. Городняя Л. В.).

Заявки присылать по адресу: 420044 Казань, ул. Х. Ямашева, д. 36. Казанский производственный комбинат программных средств.

В рамках программы создана Всесоюзная экспертная комиссия по оценке программных средств и учебно-методических материалов под председательством д-ра пед. наук А. А. Кузнецова. В комиссию входят специалисты различных сфер образования, имеющие опыт работы в области экспертизы. Информация о комиссии и результатах ее работы будет опубликована.

На базе Московского института повышения квалификации педагогических работников (бывший МГИУУ) ведутся работы по созданию телекоммуникационного центра, который, надеемся, в будущем объединит сначала республиканские ИУУ, а затем и другие учреждения народного образования. Руководит его работой Н. Д. Угринович, статья которого опубликована в этом номере журнала.

Сибирский межрегиональный фонд «Информатика: развитие и образование»

Сибирский межрегиональный фонд «Информатика: развитие и образование» (ИнфРО) создан по инициативе группы научных сотрудников Новосибирского научного центра с целью содействия информатизации школьного образования и общества в целом.

Цели и задачи деятельности фонда:

расширение круга лиц, содействующих процессу информатизации образования и общества в целом;

проведение школ юных программистов, олимпиад по программированию и других научных и культурных мероприятий;

создание программных и программно-методических средств по школьному курсу информатики и т. п., стимулирование авторов инициативных разработок;

вовлечение подрастающего поколения в процесс информатизации общества через овладение компьютерной грамотностью и основами информатики посредством его активного участия в учебно-экспериментальных разработках новых информационных технологий;

поиск и внедрение новых форм обучения на базе новых информационных технологий (телекоммуникация, аудиовидеокомпьютер и т. д.). Осуществление консультационной деятельности и информационного обслуживания;

издание научных и методических сборников, учебных пособий и других материалов;

повышение профессионального уровня научных сотрудников, участвующих в деятельности фонда; расширение их кругозора в научной, социальной, экономической сферах жизни общества.

Фонд действует на правах общественной организации с использованием спонсорских средств и целевых взносов.

Приглашаем к сотрудничеству заинтересованные организации, коллективы, а также частных лиц.

Адрес для справок: 630090 Новосибирск, пр. Акад. Лаврентьева, 6. Тел. (3832) 35-54-70, 35-03-52.

УВАЖАЕМЫЕ ЧИТАТЕЛИ!

В наши дни стало традиционным жаловаться на тяжелое финансовое положение. Тем не менее речь пойдет именно об этом.

Летом прошлого года перед подписной кампанией стоимость «ИНФО» была увеличена до 2 руб. за номер. Определяя цену, мы хотели свести к минимуму возможные убытки и в то же время не потерять подписчиков. Последнее нам удалось. А вот финансовые расчеты перечеркнуты январским резким (в 4—5 раз) ростом цен на бумагу, полиграфию и распространение. Вырученных от подписки денег достаточно лишь на выпуск двух номеров, а предполагаемые убытки превышают миллион рублей. Более того, рост тиража, как это ни абсурдно, множит убытки, поэтому с мая, возможно, будет приостановлена текущая подписка, а ее стоимость на следующий год неизбежно возрастет.

Многие издания сегодня вынуждены выпускать сдвоенные номера прежнего и даже уменьшенного объема. Такое «решение» проблемы, к сожалению, в ущерб читателям, было предложено и педагогическим журналам, с чем им пришлось согласиться.

Мы пытаемся избежать сокращения объема и периодичности «ИНФО», поэтому обращаемся к нашим коллегам и партнерам, организациям и учреждениям, центрам и ассоциациям с просьбой помочь журналу средствами. В свою очередь редакция готова предоставить площадь в журнале, а также обсудить иные формы сотрудничества.

Ждем ваших предложений.

Телефоны редакции: 261-77-81, 261-11-29. Адрес для переписки: 119034, Москва, Смоленский б-р, 4, изд-во «Педагогика», журнал «Информатика и образование».

Телефон для справок в случае переезда редакции: 246-59-69.

Редакция считает своим долгом выразить искреннюю признательность Ларисе Владимировне Ануфриенко, председателю Волгодонского областного фонда образования, за практическую помощь в приобретении бумаги.

ТЕХНОКОМ

МЫ УВЕРЕННО ПРЕДЛАГАЕМ ВАМ УЧЕБНЫЙ КОМПЬЮТЕР УКНЦ

И ВСЁ ЧТО ПОМОЖЕТ СДЕЛАТЬ ВАШУ РАБОТУ НА НЕМ БОЛЕЕ ТВОРЧЕСКОЙ, БОЛЕЕ ПРОДУКТИВНОЙ И БОЛЕЕ ПРОФЕССИОНАЛЬНОЙ ВО ВСЕХ ОТНОШЕНИЯХ!

ПОСТАВКА УЧЕБНЫХ КЛАССОВ - ВЫБОР КОНФИГУРАЦИИ В ВАШИХ РУКАХ !

КВАЛИФИЦИРОВАННАЯ ГРУППА НАШИХ СПЕЦИАЛИСТОВ ВСЕГДА ГОТОВА ПРОВЕСТИ НЕОБХОДИМОЕ ТЕХНИЧЕСКОЕ ОБСЛУЖИВАНИЕ И РЕМОНТ ЗАКАЗАННЫХ ВАМИ КЛАССОВ.

ВСПОМОГАТЕЛЬНЫЕ ТЕХНИЧЕСКИЕ СРЕДСТВА

- СТЕНД ДИАГНОСТИЧЕСКИЙ ДЛЯ РЕМОНТА КОМПЬЮТЕРОВ ТИПА УК И БК
- ЭЛЕКТРОННЫЙ ДИСК ЕМКОСТЬЮ 1 МБАЙТ
- BASIC НА КАССЕТЕ ПЗУ
- РЕДАКТОР ТЕКСТА НА КАССЕТЕ ПЗУ С ВЫВОДОМ НА ПЕЧАТЬ

ПРОГРАММНЫЕ ПРОДУКТЫ,

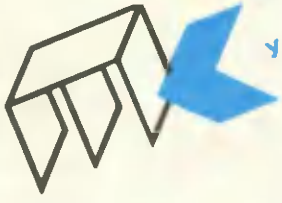
- ГРАФИЧЕСКАЯ ОБУЧАЮЩАЯ СИСТЕМА GUESS
- МАЛАЯ ИЗДАТЕЛЬСКАЯ СИСТЕМА POLYSS
- МНОГОФУНКЦИОНАЛЬНЫЙ СЕТЕВОЙ МОНИТОР NETSVD
- ФАЙЛОВЫЙ МОНИТОР SAM
- ГРАФИЧЕСКИЙ РЕДАКТОР GRUK
- УНИВЕРСАЛЬНЫЙ ДРАЙВЕР ПЕЧАТИ EP
- ПРОЦЕССОР НАСТРОЙКИ SETGEN
- ЭЛЕКТРОННАЯ ТАБЛИЦА UKCALC
- БИБЛИОТЕКА ЭКРАННЫХ ШРИФТОВ И СИМВОЛОВ MENFNT
- ИГРОВЫЕ ПРОГРАММЫ
- ИСХОДНЫЕ ТЕКСТЫ СИСТЕМЫ ПЗУ
- МНОГОКРАТНОЕ УВЕЛИЧЕНИЕ ЛОГИЧЕСКИХ И ГРАФИЧЕСКИХ ВОЗМОЖНОСТЕЙ ВАШИХ КОМПЬЮТЕРОВ!

ПРОГРАММЫ ОБУЧЕНИЯ НА КУРСАХ

- КУРСЫ ПОДГОТОВКИ ПРЕПОДАВАТЕЛЕЙ КОМПЬЮТЕРНЫХ КЛАССОВ
- КУРСЫ ПО РЕМОНТУ И ОБСЛУЖИВАНИЮ УКНЦ

БЫСТРОЕ И ЭФФЕКТИВНОЕ ОБСЛУЖИВАНИЕ ВСЕГДА БУДЕТ ПОД РУКОЙ!

У НАС ДЕЙСТВИТЕЛЬНО ЕСТЬ ВСЁ, ЧТО МЫ ПРЕДЛАГАЕМ !



ЧТОБЫ В ЭТОМ УБЕДИТЬСЯ, ОБРАТИТЕСЬ В НАШ ОФИС ПО АДРЕСУ:

103489 МОСКВА, АБОНЕНТНЫЙ ЯЩИК 51,

МНПП «ТЕХНОКОМ»,

ТЕЛ. 536-50-90, 536-50-51, 536-60-29

МАЛОЕ НАУЧНО-ПРОИЗВОДСТВЕННОЕ ПРЕДПРИЯТИЕ

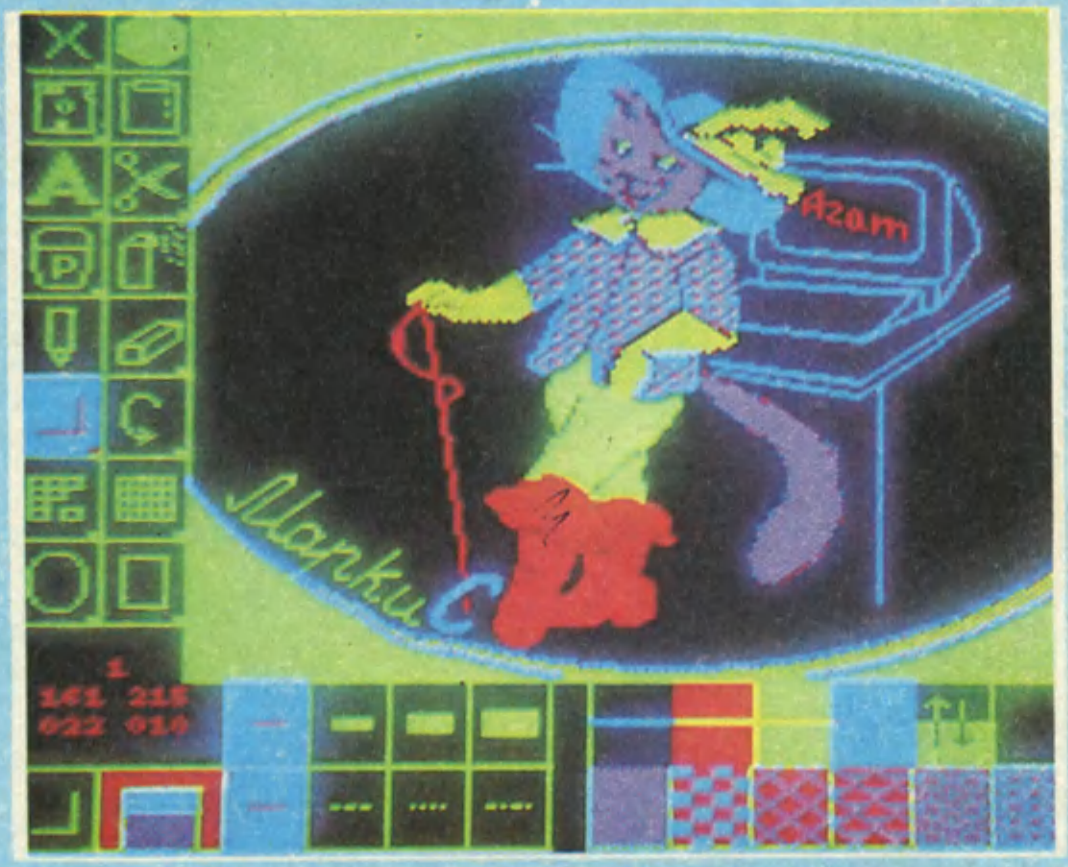
ТЕХНОКОМ

Цена 2 руб.
70423

ИНФО

2'92

ИНФОРМАТИКА И ОБРАЗОВАНИЕ



ISSN 0234—0453. Информатика и образование, 1992, № 2, 1—128.