ISSN 0234—0453

ИНФО

ИНФОРМАТИКА И ОБРАЗОВАНИЕ

1 1993

Во всем мире компьютеры фирмы Apple широко используются для целей образования на всех уровнях — от начальной до высшей школы. Это результат того, что с момента основания фирмы применение компьютеров в образовании стало одним из основных стратегических направлений ее деятельности.

И сегодня, выходя на российский рынок, Apple традиционно делает акцент на сферу образования. Фирма намерена предложить ряд комплексных решений и различные конфигурации для учебных целей.

Apple Macintosh Classic идеален для начинающих. Он содержит встроенный дисплей, флоппи-дисковод Apple Super Drive, SCSI-порт для подключения периферийных устройств (внешних жестких дисков, CD-ROM дисководов, сканеров).

Научно-методический журнал
Учрежден Министерством образования РФ и коллективом редакции

Издается с августа 1986 г.
Выходит шесть раз в год.
Москва, «Информатика и образование», 1993

ИНФОРМАТИКА И ОБРАЗОВАНИЕ

Содержание

Клуб БК

Клуб «Корвет»

Клуб УКНЦ

Педагогический опыт

Олимпиады по информатике

Информация

Обложка *С. Плюща*
Редактор отдела *Н. Луцкая*
Редактор *Н. Васильева*
Художественный редактор *Л. Розанова*
Корректор *Н. Васильева*

Сдано в набор 29.12.92. Подписано в печать 19.02.93. Формат 70×100 1/16. Бумага офсетная. Печать офсетная. Усл. печ. л. 10,4. Усл. кр.-отт. 21,8. Уч.-изд. л. 13,4. Тираж 31 630 экз. Заказ 1692. Цена 50 руб.

Почту направлять по адресу:
117485, Москва, а/я 34.
Журнал «Информатика и образование».
Телефон 208-30-78

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

Н. УГРИНОВИЧ
Московский институт повышения квалификации работников образования

Программа непрерывного курса информатики для средней школы

Программа содержит три блока: *пропедевтический, базовый* и *углубленный*. Внутри каждого блока выделены основные темы и раскрыто их содержание. Количество часов и соответственно глубина изучения тем могут варьироваться в зависимости от конкретных условий в каждой школе.

Пропедевтический курс информатики

В пропедевтическом курсе информатики учащиеся получают элементы знаний и умений по работе на компьютере, а также навыки конструирования и управления объектами (преимущественно в игровой форме). Полученные умения позволяют использовать компьютер на уроках математики, языка, изобразительного искусства и музыки для контроля знаний, тренажа и обучения.

В качестве программно-методической поддержки рекомендуется использовать Роботландию, включающую в себя программную поддержку, учебное пособие для ученика и методическое пособие для учителя (разработано для ПЭВМ «YAMAHA», УКНЦ, IBM-совместимых компьютеров). В Роботландии выделяются четыре основные педагогические направления: информационное, компьютерное, алгоритмическое и творческое. Естественно, что все они органически переплетаются в содержании и методике обучения.

1. Информация в нашей жизни

Информация вокруг нас (примеры из окружающего мира). Многообразие форм информации (рисунки, тексты, звук) и способов ее обработки. Как человек воспринимает информацию. Что можно делать с информацией. Распознавание, хранение, передача, обработка и поиск информации. Кодирование информации.

2. Компьютер как средство работы с информацией

Первичные представления о компьютере. Назначение и взаимосвязь его частей. Начальные навыки работы на клавиатуре и считывания информации с дисплея. Компьютер как универсальный исполнитель. Моделирование исполнителей (в игровой форме). Практическое использование компьютера для работы с текстом, графикой, звуком.

3. Элементы алгоритмического мышления

Понятие алгоритма и формализация записи. Исполнитель, система команд исполнителя, исполнение алгоритма. «Черные ящики» — поиск алгоритмов их действия.

4. Формирование логического мышления и творчества

Формирование логического мышления в процессе создания и выполнения алгоритмов. Навыки самостоятельного исследования, развитие интуиции, получение логических выводов (при работе с «черными ящиками»). Развитие способностей сочинения и конструирования при работе с редакторами.

Базовый курс информатики (6—9 классы)

Базовый курс информатики должен давать необходимые знания и умения будущему квалифицированному пользователю компьютера. Учащийся получит представление о возможностях компьютера и навыки его практического использования в области обработки текстов и изображений, упорядоченного хранения и поиска информации в базах данных, обработки больших массивов числовой информации в электронных таблицах.

В курсе формируется алгоритмическое

мышление и умение программировать на одном из алгоритмических языков.

Существенное место занимают вопросы решения задач на компьютере, включая этапы создания компьютерной модели объектов или процессов и проведение компьютерного эксперимента.

6 *класс*

1. Знакомство с компьютером

Общее представление о компьютере и его возможностях. Практическое знакомство с компьютером. Техника безопасности и правила работы на компьютере. Клавиатурный тренажер, игры.

Обработка текстовой информации на компьютере. Текстовый редактор. Основные возможности по созданию и редактированию текстов. Вывод текста на принтер. Запись и считывание текстов с диска.

Обработка графической информации на компьютере. Графический редактор. Система графических меню. Основные возможности по созданию и редактированию изображений. Вывод графики на принтер. Запись и считывание изображений с диска.

2. Исполнители и алгоритмы

Исполнитель. Система команд исполнителя. Понятие об алгоритме. Примеры исполнителей и алгоритмов. Способы описания алгоритмов.

Линейный алгоритм. Непосредственный и программный способ исполнения алгоритма.

Разветвляющийся алгоритм. Условие.

Циклический алгоритм. Тело цикла и условия его выполнения.

7 *класс*

3. Элементы алгоритмизации и программирования

Основные типы алгоритмов (линейный, разветвляющийся, циклический) и их реализация на языке программирования.

Линейный алгоритм. Простые вычислительные алгоритмы. Понятие переменной: имя и значение. Семантический смысл операторов (рассмотрение выполнения программы на модели компьютера).

Разветвляющийся алгоритм. Разветвляющиеся алгоритмы с символьными переменными. Программы контроля знаний по различным предметам.

Циклический алгоритм. Понятие сценария. Циклические алгоритмы и программы построения изображений. Реализация движения объектов на экране. Вложенные циклы.

4. Графическое моделирование

Графические возможности языка программирования. Графические режимы работы экрана. Основные операторы графики (возможности, формат). Основные графические примитивы (точка, линия, прямоугольник, окружность). Самостоятельная работа учащихся по созданию программ построения различных графических изображений.

Понятие модели. Объекты и их отображение на компьютере. Статические модели. Имитация движения на компьютере. Динамические модели.

Компьютерные игры. Компьютерная игра как модель воображаемого мира.

8 *класс*

5. Алгоритмизация и программирование

Вспомогательные алгоритмы. Встроенные функции, функции пользователя, подпрограммы. Сценарии сложных многоэлементных изображений и их реализация с помощью вспомогательных алгоритмов. Формальные и фактические параметры. Вызов исполнителей. «Сложные» и базовые команды исполнителей.

Алгоритмы работы с символьными переменными. Функции работы с символьными переменными («склеивание», «вырезка», длина и т. д.). Алгоритмы и программы работы с этими функциями (поиск и замена букв, слова «перевертыши» и т. д.).

Массивы. Понятие таблицы и массива. Различные типы массивов и их описание. Способы заполнения и распечатки массивов. Алгоритмы поиска информации в массивах.

Дополнительные возможности языка программирования. Функции преобразования типов величин (CHR$, ASC, VAL и т. д.). Управление движением графических объектов на экране. Функции непосредственного доступа к памяти. Отладка программ.

6. Информационные модели. Компьютерное моделирование

Этапы решения задач на компьютере. Математическая модель — алгоритм — программа. Выполнение программы на компьютере и анализ результатов. Решение задач из курсов математики и физики.

Информационная модель исследуемого объекта или процесса. Информационная модель исполнителя.

9 *класс*

7. Информация и компьютер

Информация, единицы измерения информации, представление различных типов информации (текстовой, графической, звуковой) в компьютере.

Компьютер как единый программно-аппаратный комплекс. Магистрально-модульный принцип построения компьютера. Представление о файле и операционной системе.

Внешняя память. Гибкие и жесткие диски, их информационная емкость, форматирование.

Устройства ввода информации: клавиатура, манипулятор типа «мышь», сканер. Устройства вывода информации: дисплей, принтер.

Работа с внешними устройствами: сохранение/считывание файлов с диска, ввод информации, вывод информации на дисплей или принтер.

8. Упорядоченное хранение и обработка информации. Электронные таблицы. СУБД.

Обработка числовой информации на компьютере. Электронные таблицы. Структура таблиц. Ввод чисел, формул и текстов в ячейки таблицы. Вывод фрагментов таблиц на принтер. Запись и считывание созданных таблиц с диска.

Система управления базами данных (СУБД). Упорядоченное хранение больших объемов информации в табличной форме в памяти компьютера. Структура баз данных. Заполнение баз данных и редактирование записей. Сортировка и поиск информации по заданному шаблону. Вывод баз данных на принтер. Запись и считывание баз данных с диска.

Курс углубленного изучения информатики (10—11 классы)

Курс углубленного изучения информатики является фактически курсом предпрофессиональной подготовки по использованию компьютеров в различных сферах деятельности. В зависимости от профиля школы (класса) перечень изучаемых тем, а также их объем может меняться.

В школах с углубленным изучением математики и физики целесообразно сделать упор на изучение программирования, математических методов и моделирование, в школах с экономическим уклоном — моделирование сложных многофакторных систем и линейное программирование, в гуманитарных школах — на текстовые редакторы и редакционно-издательские системы.

Тема 1. Архитектура компьютера и операционная система

Аппаратная («hardware») и программная («software») компоненты компьютерной системы. Магистрально-модульный принцип построения компьютера. Шины адреса и данных. Ячейки памяти и их адреса. Адресное пространство процессора и оперативная память. Периферийные устройства компьютера.

Файл. Файловая система. Размер файла и его размещение на диске. Имя файла. Линейные и иерархические каталоги дисков. Файловая система.

Операционная система. Назначение операционной системы. Состав операционной системы: базовые модули, драйверы внешних устройств, процессор командного языка. Загрузка операционной системы.

Команды операционной системы. Резидентные и транзитные команды. Просмотр и печать каталога. Копирование файлов. Просмотр содержимого файла. Вывод содержимого файла на печатающее устройство. Переименование файла. Удаление файлов. Создание и удаление подкаталога, вход в подкаталог и выход из него, указание маршрута поиска файла. Командные файлы.

Программные оболочки операционных систем:

NORTON COMMANDER,
DOS-SHELL,
WINDOWS.

Тема 2. Арифметические и логические основы компьютера

Арифметические основы компьютера. Позиционные системы счисления (десятичная, шестнадцатеричная). Перевод чисел из одной системы счисления в другую. Двоичная арифметика. Машинные коды чисел: прямой, обратный, дополнительный. Действия с кодами двоичных чисел. Представление в компьютере целых и вещественных чисел. Машинные единицы информации.

Логические основы компьютера. Основные понятия и область применения алгебры логики. Элементарные логические операции. Логические функции. Построение таблиц истинности. Законы (аксиомы) алгебры логики. Вычисление логических выражений при заданных наборах переменных. Упрощение логических выражений (на основе законов алгебры и их следствий).

Базовые логические элементы компьютера («И», «ИЛИ», «НЕ») и логические функции. Конструирование основных логических блоков компьютера (сумматор, триггер).

Структурная организация и функционирование компьютера. Функциональные узлы компьютера; регистры, сумматоры. Организация памяти и процессора. Понятие об арифметико-логическом устройстве и устройстве управления. Организация автоматического исполнения программы.

Тема 3. Алгоритмизация и языки программирования

Построение алгоритмов методом последовательной детализации. Основные принципы структурного программирования. Переменные в алгоритмах (числовые, символьные, логические). Алгоритмы на линейно упорядоченных множествах: поиск наибольшего (наименьшего) элемента, сортировка. Комбинаторные алгоритмы (генерация перестановок). Рекурсивные алгоритмы.

Классификация языков программирова-

ния. Машинный язык, ассемблер, языки высокого уровня. Трансляторы языков: интерпретаторы и компиляторы. Этапы работы трансляторов.

Изучение одного из процедурных языков (Паскаль, Си и т. п.). Основные типы и способы организации данных. Типы переменных. Основные операторы и конструкции языка.

Тема 4. Основы логического программирования. Язык Пролог

Процедурные и декларативные языки программирования. Язык Пролог. Базы знаний. Факты, правила, вопросы.

База знаний «Семья». Отношения (предикаты), константы и переменные. Логические связки и их реализация на Прологе. Правила де Моргана, их значимость.

База знаний «Классный журнал». Встроенные арифметические функции. Кванторы существования и всеобщности. Реализация кванторов на Прологе. Аналоги правил де Моргана для кванторов. Логический «замкнутый круг» и его процедурный аналог — «зацикливание».

Нахождение наибольшего из чисел. Принцип математической индукции. Рекурсия. Пополнение базы знаний «Семья». Вычисление факториала.

Списки. База знаний «Новый классный журнал». Операция над списками. Решение универсальных проблем обработки списков и применение результатов к решению конкретных задач.

База знаний «Железнодорожная справочная». «Интеллектуализирование» функций справочной службы. Суммирование часов и минут.

Графические средства языка Пролог. Рекурсия в графике.

База знаний «Метеорология». Направление поиска и методические принципы создания «практичных» баз знаний.

Тема 5. Решение задач на компьютере

Этапы решения задачи: построение математической модели, разработка и кодирование алгоритма, отладка программы. Проведение компьютерного эксперимента.

Графики функций и численные методы в математике. Построение графиков функций и их исследование. Решение уравнений графическим методом. Итерационные методы. Решение уравнений методом половинного деления. Вероятностные математические модели (метод Монте-Карло).

Моделирование динамических процессов. Физические процессы и их моделирование. Графическое представление физических объектов и процессов. Биологические процессы и их моделирование. Исследование развития популяций.

Имитационное моделирование сложных систем. Сложные (многопараметрические) системы и их моделирование. Методы линейного программирования. Целевая функция, поиск экстремума. Экономические модели (определение оптимальной стратегии производства изделий с учетом ограничений и т. д.).

Тема 6. Применения компьютера

Обработка текстовой информации на компьютере. Типовой набор команд редактора текстов. Считывание и запись текстового файла. Перемещение курсора по тексту. Стирание, вставка и замена символов, слов и строк. Блочные операции: выделение блока, копирование, перемещение, удаление блоков, запись блока на диск и вывод на печать. Поиск в тексте и замена. Задание формата текстового документа, форматирование. Задание системных параметров работы редактора. Перекодирование файлов. Проверка правописания. Настольные издательские системы.

Обработка графической информации на компьютере. Возможности графических редакторов. Назначение и работа с манипулятором «мышь». Создание и корректировка изображений и текста. Блочные операции: копирование, перемещение, зеркальное отображение, повороты, растяжение, сжатие, удаление и изменение фрагментов изображения. Создание зрительных и звуковых эффектов. Возможности создания мультипликации. Сканер и обработка получаемых изображений. Вывод на печать.

Обработка числовой информации на компьютере. Назначение и возможности электронных таблиц. Структура таблиц. Имя ячейки. Активная ячейка. Перемещение курсора. Строка подсказки. Строка ввода команд. Ввод текста, формул, команд и подкоманд. Формат ячейки. Изменение информации в строке ввода. Перечень и назначение основных команд. Стандартные функции.

Система управления базами данных (СУБД). Назначение системы управления базами данных. Реляционные, иерархические, сетевые базы данных. Упорядоченное хранение больших объемов информации в табличной форме в памяти компьютера. Структура баз данных. Заполнение баз данных и редактирование записей. Сортировка и поиск информации по заданному шаблону. Вывод баз данных на принтер. Статистический анализ данных и его вывод в графической форме. Запись и считывание баз данных с диска.

Интегрированные среды. Система FRAMEWORK. Понятие фрейма. Перекачка информации между различными компонентами системы. Пользовательский интерфейс.

Тема 7. Компьютерные телекоммуникационные сети

Информационные сети. Локальные, региональные, глобальные информационные сети. Назначение и возможности.

Аппаратные и программные компоненты информационной сети. Головной и абонентские пункты: требования к техническим параметрам компьютеров. Типы модемов и их характеристики. Линии связи. Протоколы связи. Сетевое программное обеспечение.

Формы использования компьютерных сетей. Организация информационных потоков в сетях. Электронная почта. Электронные конференции и электронные доски объявлений. Информационно-справочные системы. Практическая работа абонента сети.

Литература

1. Основы информатики и вычислительной техники: Учебник для средних учебных заведений. А. Г. Кушниренко и др. М.: Просвещение, 1988, 1991.

2. Основы информатики и вычислительной техники: Учебное пособие для X—XI классов средней школы. В. А. Каймин и др. М.: Просвещение, 1989, 1990.

3. Основы информатики и вычислительной техники: Пробное учебное пособие. А. Г. Гейн и др. М.: Просвещение, 1991.

4. Информатика (теория, методика, задачи). Угринович Н. Д., М.: МИПКРО, 1991.

КНИГА-ПОЧТОЙ

наложеным платежом книги по информатике

Для заведующих кабинетами информатики:

1. **Дегтярев Е.К.** *"Тенденции развития вычислительной техники"*, 64 стр., цена 50 руб. Книга незаменима при выборе нового компьютера, сети, периферийного оборудования, программ. Пачка 30 экз. 600 руб.
2. **Малютин Э.А.** *Кодовые таблицы персональных компьютеров*, 36 стр, цена 25 руб. Кодовые таблицы внутримашинного представления символов, включая псевдо-графические символы и буквы русского алфавита. Пачка 100 экз. 1500 руб.
3. **Мизрохи С.В.** *"Операционная система MS DOS"*, 64 стр., цена 40 руб. Описаны возможности операционной системы MS DOS: команды, способы построения пакетных файлов и конфигурации. Сравниваются версии 3, 4 и 5. Пачка 40 экз. 1000 руб.

Для школьников:

4. **Малютин Э.А.** *"Программирование музыкального сопровождения на Бейсике"*, 16 стр., цена 10 руб.

Для студентов и преподавателей:

5. *"Краткий справочник по языку программирования С"*, (1 сложенный лист). Цена 10 руб. Пачка 100 экз. 500 руб.
6. **Маслов А.Н.** *"Введение в язык программирования С"*, 64 стр., цена 30 руб. Пачка 80 экз. 1000 руб.
7. **Малютин Э.А., Шитов Ю.И.** *"Английский язык для программистов. Тексты и упражнения"*, 96 стр., цена 60 руб. Пачка 60 экз. 3000 руб.
8. **Дегтярев Е.К.** *"Введение в операционную систему UNIX"*, 128 стр., цена 80 руб. Пачка 30 экз. 1000 руб.
9. **Нейман В.Г.** *"Решение научных, инженерных и экономических задач с помощью пакета прикладных программ STATGRAPHICS"*, 88 стр., цена 80 руб. Пачка 50 экз. 3000 руб.

Для программистов:

10. **Стариков Ю.А.** *"Мобильность программ и особенности реализаций языка С"*, 88 стр., цена 50 руб. Пачка 60 экз. 2000 руб.
11. **Свиридов С.В.** *"Системные вызовы ОС UNIX"*, 96 стр., цена 50 руб. Пачка 60 экз. 2000 руб.
12. **Намиот Д.Е.** *"Основные возможности языка С++. Реализация Turbo С++"*, 96 стр., цена 50 руб. Пачка 60 экз. 2000 руб.
13. **Богословский А.В.** *"Системное программирование на Ассемблере для IBM-совместимых персональных компьютеров"*, 96 стр., цена 90 руб. Пачка 50 экз. 3000 руб. или 100 экз. 5500 руб.

Распространители книг приглашаются к сотрудничеству.

Заказы частных лиц и организаций направлять по *адресу:*
125438, Москва, а/я 51, Маслову А.Н.

Приложение: Образец заявки
Директору КЦО "Луч".
Прошу выслать наложеным платежом по адресу:
....
следующие книги:
далее для каждой книги указать номер по каталогу, название, стоимость, число экземпляров, цену.
Оплату книг и стоимости пересылки гарантирую.
Подпись.

Компьютерный центр обучения "Луч" приглашает **на летнюю подготовку** *преподавателей информатики школ и других средних учебных заведений, использующих IBM-совместимые компьютеры.*

Качество обучения в ЦЕНТРЕ соответствует международным стандартам и достигается за счет применения интенсивных методов обучения, постоянного тесного контакта с преподавателями (учебная группа состоит из 4-5 слушателей), индивидуальной практической работы на компьютере (по 4 часа ежедневно), ТСО, обеспечения каждого слушателя уникальной технической документацией (в личное пользование). Занятия ведут преподаватели высшей квалификации и специалисты-практики.

Темы занятий: выбор программных и технических средств для кабинета информатики; методы и средства создания обучающих компьютерных программ; программирование на языках С, Pascal или Basic по выбору; сравнение работы в MS DOS, Norton Commander и Windows; базы данных; текстовые и графические редакторы; организация предприятия с участием школьников на базе кабинета информатики; международная компьютерная терминология.
Программа занятий согласуется со слушателями.

Курс обучения - 2 недели в июле или августе 1993 года.
Размещение иногородних в гостиницах.
Стоимость обучения - 195 тысяч рублей за одного слушателя.
Заявки направлять по адресу: 125438, Москва, а/я 51, КЦО "Луч".
Тел. (095)-456-58-68.

В. ТИМАШОВ, Т. КАРАСЕВА, В. ГОМЗЯКОВА
Республиканский институт повышения квалификации работников образования

Программа курса «Основы информатики и вычислительной техники»

Интенсивное развитие процесса информатизации общества, опирающееся на широкое внедрение достижений информатики и вычислительной техники в различные сферы деятельности, ставит перед средней общеобразовательной школой задачу — подготовить учащихся к использованию полученных знаний в последующей профессиональной деятельности.

Программа состоит из трех частей:

1. Пояснительная записка.
2. Содержание обучения.
3. Примерное тематическое планирование.

Пояснительная записка

Программа рассчитана на четырехлетнее изучение курса в следующем объеме:

VIII класс (34 час. в год, 1 час в неделю);
IX класс (68 час. в год, 2 час. в неделю);
X класс (68 час. в год, 2 час. в неделю);
XI класс (34 час. в год, 1 час в неделю).

Изучение курса, предусмотренного программой, должно сформировать у учащихся знания, умения, навыки по следующим вопросам.

1. Устройство и принцип функционирования ЭВМ.
2. Освоение техники работы на ЭВМ.
3. Порядок освоения средств информационных технологий.
4. Алгоритмические структуры, построение алгоритмов.
5. Составление программ для ЭВМ по разработанным алгоритмам.
6. Этапы решения задач на ЭВМ.
7. Формализованное описание поставленной задачи, элементарные знания о методах математического моделирования.

Программа ориентирована на большой объем практических работ с использованием ЭВМ (до 50 % учебного времени) по всем изучаемым разделам. Работы с ЭВМ могут проводиться по трем формам. Это:

1. *Демонстрационная* — работу на ЭВМ выполняет учитель, а учащиеся наблюдают за ним через демонстрационный экран, либо воспроизводят действия на рабочих местах.
2. *Фронтальная* — недлительная, но синхронная работа учащихся по освоению или закреплению материала под руководством учителя.
3. *Самостоятельная* — выполнение самостоятельной работы с компьютером в пределах одного, двух или части урока. Учитель обеспечивает индивидуальный контроль за работой учащихся.

В требованиях к знаниям и умениям сделан естественный упор на знания и умения, которыми должны владеть учащиеся при изучении конкретной темы, для приобретения ими навыков работы с вычислительной техникой и освоения следующих тем курса.

Важная роль в программе отводится межпредметным связям с физикой и математикой. На них основано раскрытие основных понятий и тем.

Преподавание курса ведется по трем учебникам «Основы информатики и вычислительной техники», а именно: под редакцией Кушниренко А. Г., под редакцией Каймина В. А., под редакцией Гейна А. Г.

Программа курса имеет циклический характер. Раскрытие темы одного раздела может быть разнесено по всему курсу обучения и идти поэтапно по мере подготовки учащихся.

Раздел «Введение и первоначальное знакомство с ЭВМ» «задает» начальный уровень понятий.

Раздел «Основы компьютерной грамотности» формирует:

1) основные понятия: «Исполнитель», «Команда», «Программа»;

2) представление об ЭВМ, как об электронном устройстве (исполнителе), понимающем лишь две команды: «Включен», «Выключен». Расширение возможностей

ЭВМ, как исполнителя, осуществляется за счет программного средства, выбор которого определяется решаемой задачей;

3) представление о видах информации, обрабатываемой на ЭВМ;

4) практические навыки работы с программными средствами.

Разделы «ЭВМ как средство накопления, обработки, хранения информации» и «Логические основы ЭВМ» развивают первоначальные понятия об ЭВМ и формируют у учащихся знания:

1) об устройстве и принципах функционирования ЭВМ;

2) о том, в каком виде и каким образом информация хранится в ЭВМ.

Знания, полученные в этих разделах, помогут учащимся быстро и осознанно освоить темы: «Понятие переменной», «Типы данных», «Комаида присваивания», «Логическое присваивание», изучаемых в разделах «Основы алгоритмизации» и «Основы программирования». Знания о возможностях ЭВМ сформируют у учащихся навыки четкой формулировки поставленной задачи перед ЭВМ, грамотного анализа полученных результатов.

Изучение языка логического программирования Пролог, позволит развить у учащихся структурный подход к составлению программ, отработать основы математической логики, познакомит с понятиями, легко осваиваемыми в Прологе и используемыми в других языках программирования.

Введение в программу алгоритмического языка для записи алгоритмов предоставляет всем учащимся в равных условиях осуществлять систематическую разработку алгоритмов, отработать методы алгоритмизации вне зависимости от типа конкретной ЭВМ и, вместе с тем, позволит осуществить достаточно легкий переход к построению программ на конкретном языке программирования.

При изучении основ программирования не делается ставка на конкретный язык программирования, учитель вправе выбирать его сам в зависимости от вида предоставленной ему техники.

В разделе «Решение задач на ЭВМ» дается поэтапный подход к решению задач, с использованием задач из других предметов школьного курса и задач по интересам учащихся, задач моделирования физических, химических, биологических и т. д. процессов с использованием несложных численных методов.

Раздел «ЭВМ в обществе» рассматривается в конце курса IX класса и в XI-м классе, как формирующий профессиональную ориентацию выпускников. Как результат изучения курса, в XI классе учащиеся выполняют небольшую индивидуальную или коллективную (2, 3 чел.) работу, по темам предложенным учителем или выбранным самостоятельно.

По окончании IX-ого класса учащиеся получают знания об устройстве и принципах функционирования ЭВМ, о типах информации, обрабатываемой на ЭВМ; об ЭВМ, как средстве использования информационных технологий; основах логического программирования; основах алгоритмизации; о принципах постановки задачи, о началах программирования.

Содержание обучения

Введение. Первоначальное знакомство с ЭВМ

Понятие об информации. Виды информации. Способы поиска, обработки, хранения информации. Общий вид ЭВМ. Составные части компьютера. Их назначение, взаимосвязь. Что изучает информатика. Информатика как наука. Определение информатики. Роль информатики и вычислительной техники на современном этапе развития общества. Локальная сеть, ее назначение. Правила техники безопасности при работе на ЭВМ. Знакомство с клавиатурой ЭВМ.

Фронтальные работы. Игровая программа, начальные навыки работы на клавиатуре ЭВМ — обучающая программа «Клавиатурный тренажер».

Самостоятельная работа: проверка навыка работы на клавиатуре ЭВМ.

Требования к знаниям и умениям.

Учащиеся должны знать: что такое информатика, названия и назначение основных узлов ЭВМ, правила техники безопасности.

Учащиеся должны уметь: приводить примеры накопления, хранения, обработки информации из профессиональной и повседневной деятельности человека, понимать роль информатики в современном обществе, включать, выключать ЭВМ, уметь пользоваться клавиатурой ЭВМ.

Основы компьютерной грамотности

Введение понятий: «Исполнитель», «Команда», «Программа».

Знакомство с компьютером как с инструментом: а) подготовки текста, графики, музыки; б) хранения, обработки, поиска информации; в) выполнения вычислений.

Фронтальные работы:

а) с инструментальными программными средствами: «Текстовой редактор», «Графический редактор», «Музыкальный редактор»;

б) с информационно-поисковыми системами типа «Базы данных»;

в) с использованием ЭВМ в режиме непосредственных вычислений, с программным средством «Электронные таблицы».

Самостоятельная работа: для закрепления и отработки практических навыков выполняются индивидуальные задания по каждому из видов информационно-поисковых и инструментальных средств.

Требования к знаниям и умениям.

Учащиеся должны знать: назначение рассмотренных программных средств.

Учащиеся должны уметь:

приводить примеры исполнителей и их команд, пользоваться «меню» и справочными средствами;

набрать текст, отредактировать и записать, вывести на экран или печатающее устройство;

построить примитивное графическое изображение на экране, вывести его на печатающее устройство, записать;

вывести на экран, отредактировать, записать ранее созданный текст или графическое изображение;

создавать новые, пользоваться уже созданными таблицами и выполнять поиск информации в базах данных;

пользоваться электронными таблицами;

иметь представление о музыкальных возможностях ЭВМ и средствах редактирования музыкальных композиций;

использовать ЭВМ в режиме непосредственных вычислений.

ЭВМ — средство накопления, хранения, обработки информации

ЭВМ — электронное устройство обработки информации. Виды и свойства информации. Системы счисления. Кодирование информации. Двоичное кодирование. Определение объема информации. Архитектура ЭВМ. Функциональная организация ЭВМ. Организационная структура оперативной памяти. Понятие ячейки, ее свойства. Принцип адресности. Принцип программного управления. Принцип хранимой программы и данных. Формы представления целых, вещественных, символьных данных в ЭВМ. Средства длительного хранения информации. Понятие файла. Первоначальные сведения о назначении операционной системы. Классификация программных средств.

Фронтальные работы проводятся с программой по обучению и контролю знаний по системам счисления; демонстрационной программой по функциональному устройству ЭВМ; демонстрационной программой по операционной системе.

Требования к знаниям и умениям

Учащиеся должны знать: способы записи чисел в двоичной системе счисления, способы перевода из десятичной системы счисления в двоичную; особенности двоичной системы счисления, назначение основных узлов ЭВМ; единицы измерения объема информации.

Учащиеся должны уметь объяснить: функциональную организацию ЭВМ (по плакату), организационную структуру оперативной памяти, назначение ячейки памяти, назначение операционной системы, формы представления данных в ЭВМ, понятие файла.

Логические основы ЭВМ

Введение в логику. Суждения. Рассуждения. Элементы формальной логики: понятия, высказывания, сложные высказывания, логические связки. Математическая логика: символьное обозначение высказываний, конъюнкция, дизъюнкция, импликация, отрицание. Составление логических выражений. Законы математической логики. Таблицы истинности. Использование математической логики в технике. Логические схемы (контактные, на полупроводниках). Упрощение логических выражений на основании законов математической логики.

Фронтальная работа по программе логические элементы, тестирование логических схем на Прологе.

Требования к знаниям и умениям.

Учащиеся должны знать: основные функции математической логики, законы математической логики.

Учащиеся должны уметь: читать логические схемы, записывать логические выражения, преобразовывать их, составлять таблицы истинности.

Логическое программирование

Знакомство с языком логического программирования Пролог. Понятие базы знаний, фактов и вопросов в языке Пролог. Правила логического вывода. Поиск ответа в базе. Понятие переменной (параметра). Составление таблиц поиска. Графические средства Пролога. Организация рекурсий. Введение параметра-счетчика. Рекурсивное задание числовых функций.

Фронтальная работа по созданию простейших программ на Прологе (формирование базы знаний, составление вопросов, анализ ответов).

Самостоятельная работа: разработка несложных индивидуальных программ на Прологе по темам из других предметов школьного курса, задачам по интересам.

Требования к знаниям и умениям.

Учащиеся должны знать: понятие факта, правила, вопроса.

Учащиеся должны уметь: формировать базу знаний, формулировать логически завершенные правила, формулировать вопросы, анализировать ответы.

Основы алгоритмизации

Знакомство с исполнителями. Система команд исполнителей. Понятие алгоритма. Свойства алгоритма. Алгоритмический язык как средство записи алгоритмов. Правила записи алгоритмов на алгоритмическом языке. Последовательное выполнение команд. Команды ветвления и повторения. Вспомогательные алгоритмы. Алгоритмы обработки данных. Понятие переменной (имя, значение). Типы переменных: числовые, текстовые, табличные. Действия над данными. Отношения между данными в качестве условий. Аргументы и результаты алгоритмов. Условие задачи, математические способы ее решения.

Фронтальные работы: знакомство с командами конкретного исполнителя, построение и исполнение алгоритма для заданного исполнителя, запись и выполнение основных конструкций алгоритмического языка; построение алгоритмов работы с данными, построение и исполнение алгоритмов решения математических задач, построение и исполнение алгоритма с использованием вспомогательных алгоритмов.

Самостоятельные работы: выполнение работы с использованием основных конструкций алгоритмического языка, вспомогательных алгоритмов.

Требования к знаниям и умениям.

Учащиеся должны знать: понятие алгоритма, его свойства, правила записи основных алгоритмических структур, типы величин, способы их описания, правила исполнения алгоритмов.

Учащиеся должны уметь: записывать на алгоритмическом языке несложные алгоритмы.

Основы программирования

Основные команды операционной системы. Назначение операционной системы. Понятие файла. Вывод каталога. Запись, считывание, удаление файла. Запуск программ. Назначение основных функциональных клавиш. Этапы обработки программ. Понятие об интерпретаторах и компиляторах. Сообщения об ошибках. Начала программирования. Назначение языков программирования. Алфавит выбранного языка программирования. Типы переменных. Арифметические операции. Операции присваивания. Библиотека встроенных функций. Логические операции. Редактирование программ. Классификация ошибок.

Фронтальные работы: реализация конструкций алгоритмического языка на языке программирования, вызов библиотечных подпрограмм, ввод и вывод данных, составление и отладка программ.

Самостоятельная работа: итоговая работа по программированию.

Требования к знаниям и умениям.

Учащиеся должны знать: запись алгоритмических структур на язык программирования, правила исполнения программ.

Учащиеся должны уметь: записать, отладить, исполнить простую программу на языке программирования, использовать стандартные функции и подпрограммы, организовать выдачу результатов исполнения программ на экран и печатающее устройство.

Этапы решения задач на ЭВМ

Постановка задачи. Построение математической модели, выбор метода решения, разработка сценария, разработка алгоритма по методу пошаговой детализации, анализ правильности алгоритмов, разработка тестов испытания программы, кодирование программы на язык программирования, составление плана ввода и отладки программы, анализ конечных и промежуточных результатов.

Фронтальные работы: решение задач из школьного курса, задач по интересам.

Самостоятельные работы: итоговое индивидуальное задание по решению задач.

Требования к знаниям и умениям.

Учащиеся должны знать: названия и содержание основных этапов решения задач.

Учащиеся должны уметь: построить простую математическую модель по конкретной задаче, разработать и исполнить программу решения задачи на основе построенной модели.

ЭВМ в обществе

Краткая история развития вычислительной техники. Представления о развитии элементной базы. Поколения ЭВМ. Их принципиальные различия. Дополнительные сведения об устройстве ЭВМ. Представление о развитии программного обеспечения. ЭВМ в производстве, управлении, проектировании, делопроизводстве, культуре, медицине, быту. Роль ЭВМ в развитии современного об-

щества. Роль компьютерной грамотности и информационной культуры в освоении новых технологий, развитии производства.

Фронтальная работа: с пакетами программ, демонстрирующими применение ЭВМ.

Требования к знаниям и умениям.

Учащиеся должны знать: поколения ЭВМ, их принципиальное различие.

Учащиеся должны уметь: приводить примеры использования вычислительной техники в областях народного хозяйства и повседневной деятельности.

Продолжение следует

Использование игровых моментов на уроках информатики в V классе

Во втором полугодии ученики 5 класса уже знают некоторые понятия информатики. Они умеют составлять простейшие алгоритмы, писать программы на естественном языке для различных исполнителей, могут разобраться в логике несложных программ. С III четверти мы переходим к изучению микрокалькулятора (МКШ-2). Этот выбор сделан неслучайно. У многих дома имеются микрокалькуляторы, но далеко не все знают их возможности и различные сервисные устройства. Чаще всего в работе пользуются только знаки арифметических действий, а память, автоматический сброс, сохранение предыдущего числа остаются в стороне.

Во втором полугодии на уроках математики закладываются такие важные понятия, как десятичные дроби и правила вычислений с ними, а ученики могут подкрепить свои знания, работая на микрокалькуляторе. Игра «Эстафета», которую мы проводим в конце учебного года, посвящена именно этому. Ее суть заключается в следующем.

Ученикам раздаются карточки-задания с примерами. Так как в кабинете имеется два ряда столов, то класс делится на две команды. Учащиеся, которые сидят за первыми партами, начинают игру. Они решают примеры и передают своеобразный «маршрутный лист» с ответом на следующую парту, а там не могут начать решение своего примера, пока не получат предыдущий результат. Ответ с последней парты передается учителю. В ходе этой игры можно заметить, что некоторые ученики готовятся заранее, вычисляя промежуточные результаты, и им остается выполнить только одно действие. Все это экономит время и дает возможность выиграть. Например, в карточке 2 можно заранее выполнить сложение в скобках, умножение и занести ответ в память. Получив «маршрутный лист» с предыдущим ответом, остается только выполнить вычитание. Например:

1. 4,3 · (26 + 17,6) — 0,8
2. — 5,58 · (2,7 + 4,71)
3. — 0,3322 + 6,25 : 0,5
4. + 2,5 · (4 — 3,9)
5. + 0,25 — 20 · (7,5 : 75)
6. — 30 · (5 + 0,2)

Примеры подобраны так, что в ответе получается число нуль. Проанализировав ответы, можно сделать вывод, на какой парте была допущена ошибка. Пострадает вся команда, так как из-за одной ошибки все дальнейшие вычисления будут неправильными.

Особо можно отметить учащихся, которые наиболее рационально подошли к решению примеров (это видно в ходе эстафеты).

Игра проходит весело, интересно, с настоящим спортивным азартом. Она также дает возможность проверить, насколько бегло ученики могут выполнять действия на микрокалькуляторе, их вычислительные навыки. После игры производится подробный анализ с обязательным разбором ошибок.

К каждому варианту примеров прилагается листок с ответами, чтобы было легко найти ошибку, сравнивая правильные ответы с ответами на «маршрутном листе».

А. Гейдар,
М. Ташбалтаев

А. ЩЕГОЛЕВ

Информатика и диалектика

Время подвигает нас к переосмыслению того, что мы делаем, заставляет задуматься о судьбах школьного курса информатики. Что нового способен привнести он в обучение? Какую воспитательную «нагрузку» способен нести? В каком направлении развивается? Для чего он, по большому счету, нужен?

В наше время кризисов и глубоких потрясений ответственность всех причастных к школе людей заключается в готовности «вооружить» вступающих в самостоятельную жизнь выпускников методами решения проблем, возникающих в их жизни. Проблемы эти подчас необычайно сложны; однако, уходя от их решения в область «чисто школьных» задач, не уподобляемся ли мы человеку, который ищет потерянную монету не там, где он ее потерял, а там, где светлее?

Ниже приводится программа учебного предмета «Информатика и моделирование процессов в природе и обществе» для 6—9 классов средней общеобразовательной школы, основной целью которого является овладение учащимися методами, позволяющими ставить и решать реальные проблемы, окружающие нас в этой жизни. Этот предмет вот уже четвертый год апробируется в ряде школ страны, работающих по модели «Экология и диалектика». За это время сложился небольшой коллектив учителей, интересующихся методологическими проблемами, переосмыслением целей, задач и организационно-методических принципов преподавания информатики. Начат выпуск учебников по данному предмету; первый из них (для 6 класса) увидел свет в конце минувшего года.

6 КЛАСС

Предельно обобщая направленность курса информатики, можно сформулировать цель обучения информатике как овладение *методологией приобретения необходимой информации об окружающем мире и самом себе*. Курс 6-го класса является вводной ступенью и решает три основные задачи:

знакомство с понятием информации;

овладение компьютерной грамотностью, т. е. умением *использовать компьютер как инструмент творческой деятельности* при подготовке текстов, рисовании, поиске информации и т. п.;

ознакомление с диалектикой необходимого и случайного. Этот раздел позволяет рассмотреть один из аспектов проблемы *творческого и нетворческого*. При наличии соответствующего компьютерного практикума соотнесение закономерных и вероятных явлений позволяет продвинуться в понимании сущности творческого в деятельности человека.

Программа изложена в форме *одного из возможных вариантов* поурочного планирования. Лабораторный практикум по курсу включает в себя работу с *программным обеспечением* двух типов. Это, во-первых, программы по теме «компьютерная грамотность». Они создаются профессионалами и включают в себя клавиатурный тренажер, редактор текстов, графический редактор, учебную СУБД.

Ко второму типу программного обеспечения относятся так называемые *«малые программы»*. Их отличительная особенность заключается в том, что они могут быть созданы самими преподавателями по сценариям или алгоритмам этих программ. При всех очевидных издержках такой подход имеет и свои положительные качества, позволяя учителю в процессе работы над программой глубже осмысливать нетривиальные закономерности, постигаемые школьниками, а также практиковаться в алгоритмизации и программировании.

В настоящее время создаются специализированные «малые программы» для разных типов учебной вычислительной техники.

ПОУРОЧНОЕ ПЛАНИРОВАНИЕ (1 ч×34 нед=34 ч)

Уроки 1, 2. Мы и информация

Знакомство с классом. «Информация всегда о чем-то». Источники информации: газеты, книги (в том числе фантастические). Зачем нужна информация (обсуждение)? Информация отражает окружающий нас мир (реальный или воображаемый). В каком виде представляется информация? (Зарубки на дереве, барабанный бой, мимика, жесты и т. п.). Подчеркиваем различие между *информацией и ее носителем*. Информация *не материальна*, ее нельзя «потрогать»!

Количество информации. Мало, много, отсутствие информации, исчерпывающая информация. Существенно зависит от того, *кто принимает сведения* (от новизны, заинтересованности). Как измерять информацию (проблема)?

Игра «Угадай номер квартиры». Задавая вопросы, выяснить, в какой квартире (из 8-ми, 16-ти и т. д.) живет отвечающий. Характер вопросов: допускаются лишь ответы *да* или *нет*. «Наилучшая» стратегия отгадывания — деление пополам. Почему пополам (проблема)?

Задача о взвешивании: с помощью чашечных весов без гирь определить, какая из 9 (27) монет является фальшивой (более легкой).

Урок 3. Знакомство с компьютерами

Ознакомление с техникой безопасности и правилами поведения в компьютерном классе. Первоначальное знакомство с клавиатурой ЭВМ. *Работа с программой «клавиатурный тренажер».*

Урок 4. Компьютер-фальшивомонетчик

Напоминание о проблемах отгадывания номера квартиры и задачи о взвешивании. Работа с *программами «Угадай номер квартиры»* и *«Найди фальшивую монету».* Эти программы «загадывают» соответственно номер квартиры или монеты, учащемуся же предлагается несколько раз за наименьшее число попыток определить эти номера. Компьютером подсчитывается количество попыток.

Проблема: почему в первом случае «оптимальная» стратегия состоит в том, чтобы делить все пространство исходов *на две*, а во втором случае — *на три* равные части? В ходе дискуссии дети должны прийти к выводу: в первом случае вопрос предполагает *два*, а во втором случае — *три* варианта ответа.

Урок 5. Я знаю клавиатуру ЭВМ

Работа с *клавиатурным тренажером.* В силу исключительной важности приобретения навыков работы на клавиатуре ЭВМ этот урок целиком посвящается углублению и закреплению этих навыков. Особое внимание следует обратить на учащихся, у которых возникают трудности при работе за компьютером.

Урок 6. Что такое текстовый редактор?

Вводное объяснение о хранении текстов во внутренней и внешней памяти ЭВМ. Работа с имеющимся в наличии *редактором текстов.* Необходимо ознакомить учащихся лишь с самыми основными возможностями программ такого типа. Команды редактора записываются в тетради.

Урок 7. Вводим текст

Работа с редактором текстов. Каждый ученик получает карточку с заданием примерно следующего вида:

Вариант 6
<Текст (стихи или проза), 8—16 строк>
Задание:
1. Ввести текст в ЭВМ
2. Исправить грамматические ошибки.
3. Распечатать строки «лесенкой».

В конце урока — контроль. Необходимо оценить качество выполнения задания (естественно, и грамотность в том числе!).

Урок 8. Выпускаем газету

Объяснение условий. Выборы (назначение) главного редактора газеты (учителя или достаточно продвинутого ученика), редакторов, корректоров, художников; определение авторских коллективов и тематики статей.

Уроки 9, 10. За работу!

Ввод подготовленных дома статей в ЭВМ при помощи текстового редактора. Распечатка текстов статей.

Урок 11. Анализ работы по выпуску газеты

Детальный анализ отдельных материалов и газеты в целом. Возможен конкурс газет между классами, либо между половинами одного класса. Критерии: оригинальность, раскрытие тем, грамотность, качество оформления. Желательно привлечение «независимых» экспертов (учеников старших классов, учителей, родителей).

Урок 12. Знакомство с графическим редактором

Освоение (и запись в тетради) основных возможностей имеющегося в наличии графического редактора.

Уроки 13, 14. Рисование на заданную тему

С помощью графического редактора по предложенному преподавателем эскизу строится изображение на экране ЭВМ. Эскиз может быть представлен на доске, либо — в нескольких вариантах — на индивидуальных карточках.

Урок 15. Самостоятельное творчество

Рисунок на экране ЭВМ по собственному замыслу и эскизу ученика.

Урок 16. Конкурс рисунков

В торжественной, праздничной обстановке проводится конкурс рисунков учеников. Критерии оценки работ — преимущественно художественные. Целесообразно создание «независимого» жюри из старших школьников, родителей, художников-профессионалов.

Урок 17. Резерв времени I полугодия

Урок 18. Что такое вероятность?

На примере бросания монеты и игральной кости (кубика) на интуитивном уровне обсуждается понятие вероятности, смысл

слов «вероятность», «шанс» и т. п. Обращается внимание на то, что эта количественная характеристика, определяемая при многократном повторении эксперимента, приобретает вполне определенный практический смысл и для однократной реализации события.

Урок 19. Закономерна ли случайность?

Обсуждение понятий закономерного (необходимого) и вероятного (случайного). Проблема: может ли из хаоса родиться закономерная структура? Определение числа «пи» методом Монте-Карло (объяснение).

Урок 20. «Пи» из песчинок

Работа с программой «Монте-Карло». На глазах у школьников из хаоса разбрасываемых точек в динамике уточняется (до определенного предела) число 3,14...

Уроки 21, 22. ЭВМ — калькулятор

Синхронная лабораторная работа непосредственно в среде интерпретатора имеющегося на ЭВМ языка (Бейсик, Фокал). Ученики проделывают предложенные учителем вычислительные действия и заносят в тетрадь в бланк определенной формы.

Урок 23. Неравномерные вероятности

Обсуждение проблемы неодинаковых вероятностей различных исходов (например, в дискретном случае — кубик со смещенным центром тяжести). Что вообще означает слово симметрия? Симметрия — это инвариантность (неизменяемость) при каком-либо преобразовании.

Сумма вероятностей всех исходов случайных событий.

Урок 24. Бросаем несимметричный кубик

Работа с программой «Несимметричный кубик». Ставится обратная задача (задача математической статистики): по результатам многократного «бросания кубика» определить вероятность каждого исхода.

Урок 25. Компьютер — творец (?)

Обсуждение «творческих» возможностей компьютера. Проблема: что есть творчество? Компьютер-поэт, компьютер-художник, компьютер-композитор...

Мы реализуем компьютерное «творчество» с помощью «генератора случайных (точнее, псевдослучайных) событий». Сущность компьютерного «творчества» — случайный подбор слов, знаков, нот и т. п. в рамках более или менее жесткой системы ограничений (системы отбора результатов), удачный выбор которой определяет «стиль» подобного творчества.

Возможны ли интересные находки? Безусловно; случай — мощное и достаточно универсальное средство для создания нового. Однако расплата за подобную «универсальность» — крайне редкое получение действительно интересных результатов; притом тем реже, чем «утонченнее» наши собственные «творческие» критерии оценивания и отбора. А отделять «творческие» результаты от всех прочих приходится «вручную».

Важный вывод: человек творит все—так и не так!

Результат как творческой, так и нетворческой деятельности — закономерен, процесс же творчества быть закономерным «не обязан».

Урок 26. Компьютер — художник

Построение псевдослучайных точек (программа «Звездное небо»), случайное блуждание (программа «Отрезки»), построение псевдослучайных прямоугольников, окружностей и т. п.

Урок 27. «Черные дыры»

Неравномерное распределение псевдослучайных точек (программа «Звездное небо»). Изменяя параметры распределения, ученики должны получить на экране изображение по предложенному учителем эскизу.

Урок 28. Компьютер — музыкант

Равномерное и неравномерное распределение вероятностей появления отдельных нот. Работа с программой «Какофония».

Урок 29. Информация хранится в ЭВМ

Знакомство с основными возможностями имеющейся в наличии системы управления базами данных. Рассматриваются лишь основные команды СУБД.

Урок 30. Работаем с базой данных

Учителем предлагается пример наполнения базы данных (один на всех либо по индивидуальным карточкам-заданиям). Ученикам необходимо с помощью имеющейся СУБД создать базу данных на ЭВМ.

Уроки 31, 32. Подведение итогов года

Прием индивидуального зачета

Уроки 33, 34. Резерв времени

ИНВАРИАНТЫ УСПЕШНОГО ЗАВЕРШЕНИЯ КУРСА 6 КЛАССА

К концу обучения в 6 классе учащиеся должны обладать следующими представлениями, закрепленными в практической деятельности (по разделам).

1. Интуитивное представление об информации

Понятие информации знакомо почти всем, однако трудно определяемо. Можно лишь противопоставить (в единстве) нематериальный характер информации и материальную сущность «отражаемых» информацией объектов.

Способ измерения количества информации допускает различные единицы ее изме-

рения. Недостаточная практичность «придуманного» способа не позволяет использовать его в большинстве реальных ситуаций.

Количество и самый факт передачи информации зависят не только от того, сколько тех или иных «единиц» информации вложил в сообщение тот, кто его передает, сколько от свойств принимающего это сообщение (насколько новым и интересным оно для него оказалось).

2. «Компьютерная грамотность»

Попытка подхода к проблеме «творчество и компьютер»: компьютер — инструмент творчества! Компьютер способен «взять на себя» рутинную, самую «неблагодарную» часть творческой работы. Компьютер — инструмент писателя, художника, ученого, библиотекаря.

Включение и выключение компьютера. Основные (минимальные) возможности характерных типов деловых программных средств: редактора текстов, графического редактора, электронной таблицы (при ее отсутствии — вычислительные возможности компьютера в режиме непосредственного исполнения директив), системы управления базами данных. Сопоставление компьютера с «традиционными» инструментами (средствами) соответствующей творческой деятельности.

Компьютер в этом качестве имеет как сильные, так и слабые стороны. Однако применение даже несовершенного средства творчества вовсе не всегда приводит к убогим с творческой точки зрения результатам. Как правило, использование менее совершенного инструмента приводит к появлению нового жанра в данном виде творчества. Именно поэтому человек имеет шанс «открыть себя» заново всего лишь при смене инструментария творческой деятельности.

3. «Творчество с точки зрения взаимосвязи необходимого и случайного»

Попытка «научить» компьютер самостоятельному «творчеству». В качестве источника творчества выступает генератор случайных событий. При многократном повторении случайных (в действительности, псевдослучайных) событий устойчиво возникают необходимые (закономерные) результаты. При этом то, что эти результаты возникают с необходимостью, вовсе не противоречит возможности появления разных вариантов этих результатов. При «случайном» творчестве вполне возможно (и даже вполне закономерно) появление действительно творческих (с «человеческой» точки зрения) результатов. Таким образом, случай выступает в качестве мощнейшего созидательного средства. Однако расплата за эту «мощь» — огромное количество «мусора», из которого желанный творческий результат приходится извлекать «вручную».

Литература

По 6 классу

1. *С. Пейперт* «Переворот в сознании. Дети, компьютеры и плодотворные идеи», М., «Педагогика», 1989.

2. *А. Г. Щеголоев* «Концептуальные основания преподавания предмета «Информатика и моделирование процессов в природе и обществе» (модель «Экология и диалектика»)», в сб. «Педагогический эксперимент «Экология и диалектика», методические рекомендации», выпуск 4, М., МГИУУ, 1990.

3. *Л. В. Тарасов* «Мир, построенный на вероятности», М., «Просвещение», 1984.

4. *Г. Секей* «Парадоксы в теории вероятностей в математической статистике», М., «Мир», 1990.

5. *Б. А. Покровский* «Введение в оперную режиссуру», М., ГИТИС, 1985.

7 КЛАСС

Основной дидактической целью курса информатики 7 класса является построение формально-логической модели мышления человека. Компьютерный практикум, которым буквально пронизан курс, заключается в модификации и построении баз знаний в среде языка Пролог. Интерпретатор одной из версий языка Пролог (например, Пролог-Д) — это единственное программное средство, которое потребуется для преподавания информатики в 7-м классе.

Язык логического программирования Пролог в процессе обучения реализует следующие функции:

по существу, является процедурным интерпретатором логических формализмов; тем самым позволяет установить взаимосвязь между декларативным представлением знаний и процедурой формального рассуждения. Это соответствие является одним из проявлений диалектики исторического и логического;

позволяет строить определенного уровня модели «внутреннего представления» человеческих знаний;

выводит на качественно новый уровень понимание возможностей использования компьютеров. ЭВМ рассматривается в качестве интеллектуального помощника, способного «брать на себя» рутинную часть мыслительной работы (проведение рутинных рассуждений).

Центральным моментом в программе этого курса является рассмотрение диалектической пары способов рассуждения: индукции и дедукции. Рассматриваемый на наглядных примерах принцип математической индукции возникает как дедуктивное обобщение индуктивного метода и подкрепляет-

ся рекурсивным лабораторным практикумом на Прологе.

При рассмотрении конкретных баз знаний (на доске и на компьютере) необходимо в каждом (!) случае добиваться от учеников полного понимания того, как именно будет осуществляться поиск ответа на данный вопрос. Следует «проигрывать» процедуру вывода, начиная с самых простых примеров. Неучет процедурной семантики Пролога ставит под угрозу достижение целей курса.

Следует особо заострить внимание учащихся на том, что общение с интеллектуальным помощником требует четкой и однозначной постановки каждой решаемой задачи.

ТЕМАТИЧЕСКОЕ ПЛАНИРОВАНИЕ (1 ч×34 нед=34 ч)

1. «Первый этаж» (3 ч)

«Воспоминания» о понятии информации. «Непрактичность» этого понятия, существенно зависящего от свойств человека, принимающего сообщения. Предложение построить модель мышления человека. Как отражается мир «в голове» человека (проблема)?

Факты — нижний уровень («первый этаж») знаний. Запись фактов на Прологе. Создание фактической части базы знаний «Семья» (по существу, пока лишь базы данных). Вопросы к базе знаний — вопросы о фактах.

Рефлексия: структура факта. Конкретные параметры и взаимосвязь (отношение) между ними. Следует минимизировать количество различных отношений; для этого нужно стремиться выражать одни отношения через другие.

Примечание. Целесообразно выписывать на доске общие «шаблоны», по которым будут составляться конкретные факты, например:

мама (<кто>, <чья>).

Это поможет ученикам сосредоточиться на сущности формализации отношений.

2. «Второй этаж» (4 ч)

Введение новых понятий в базу знаний. Необходимость во введении общих правил. «Определение» бабушки. Структура правила: левая часть (определяемое), «стрелочка» и правая часть.

Благодаря правилам компьютер приобретает возможность рассуждать! Процедура поиска ответа на конкретный вопрос. Как рассуждает человек? Примеры из математики, других наук, «житейские» примеры. Изъяны (дефекты) в рассуждениях. Понятие о логике.

Смысл знака «,» в правой части правила. Логические связки и, или. Уточненное «определение» бабушки. Реализация логических связок на Прологе.

Отрицание. Логическая связка не. Реализация на Прологе. Закон двойного отрицания.

Взаимодополняющий характер и проблема взаимозамены связок и и или. Правила де Моргана.

Определение понятия «родитель» с использованием логической связки или.

Определение понятий «брат», «сестра», «дядя», «тетя» и т. п. с использованием логических связок.

Рефлексия: база знаний представляет собой как бы двухэтажное здание, нижний этаж которого («Что?») составляют факты, а верхний («Как?») — правила. Вопросы в Прологе всегда задаются о фактах; компьютер же либо находит факт (что нужно ответить), либо правила (как из имеющихся фактов с помощью рассуждений вывести ответ на вопрос).

3. Рекурсия и индукция (6 ч)

Необходимость пополнения базы знаний «Семья», понятие предка. Родители, дедушки, бабушки, прадедушки, прабабушки...— все это предки. Индукция (в буквальном переводе «наведение») — рассмотрение частных случаев с целью обобщения, в определенном смысле путь «снизу вверх» (от частного к общему).

Как записать «определение предка» в общем виде? Неполная индукция уже не поможет, так как необходимо определить это отношение на произвольную «глубину».

Можно предложить следующий вариант определения предка:

1. Родители — это предки.
2. Родители предков — тоже предки.

Запись на Прологе. Проверка на компьютере. Детальное рассмотрение процедуры выполнения!

Почему именно факт и правило, какова их роль? Принцип математической индукции (на специально подобранных примерах, минимум формул), например:

что можно сказать об очереди, в которой за каждой женщиной обязательно стоит женщина? Демонстрируется роль базиса индукции;

«доказательство» того, что во всякий автобус может «войти» любое наперед заданное количество людей;

доказательство того, что любую сумму в целое число рублей, начиная с восьми, можно выплатить, располагая только «трешками» и «пятерками».

4. Рекурсивный практикум (3 ч)

Логические операции на Прологе. Запись на Прологе суммирования последовательных

натуральных чисел. Роль рекурсии; процедура выполнения!

Целесообразно проведение самостоятельной работы. Варианты заданий: суммирование последовательных четных (нечетных) натуральных чисел и т. п. Отдельным ученикам можно предложить задачи на умножение (факториал и т. п.), однако это сопряжено с быстрым ростом результата в зависимости от аргумента и превышением максимального целого числа, представимого в данной ЭВМ (что, конечно же, тоже может стать предметом обсуждения). Наиболее сильным ученикам полезно решать задачи на делимость, обнаружение простых чисел и т. д.

5. Что значит «от противного»? (5 ч)

Понятие прямой, обратной, противоположной и противоположной обратной теорем. Истинность и ложность этих утверждений друг относительно друга — формальный подход (поскольку мы делаем выводы об истинности суждений, не вникая в их содержание).

«Направление» рассуждения. Теоремы — это правила! Практикум на Прологе.

Сущность доказательства утверждений от противного. Формальные ошибки в рассуждениях: подмена прямого утверждения обратным или противоположным. Примеры.

6. «Снизу вверх» и «сверху вниз» (4 ч)

Индукция и дедукция — два взаимодополняющих способа рассуждений. Их противоположность и единство. Роль: в математике, в философии, в частных науках. Исторический экскурс.

«Дедуктивный метод» Шерлока Холмса. Только ли «дедуктивный»? Что такое доказательность? Что такое научность? Безмашинный практикум: большое количество примеров рассуждений. Обратить особое внимание на самостоятельные рассуждения школьников. Возможна работа по индивидуальным заданиям.

Новый взгляд на проблему творческой деятельности сквозь призму данной категориальной пары (индукция и дедукция). Разница в процессе получения дедуктивных результатов компьютером, отвечающим на вопрос к базе знаний, и человеком. Компьютер осуществляет исчерпывающий перебор всевозможных фактов и правил (очевидно, это нетворческий процесс). Человек же обладает способностью индуктивного (творческого) отбора необходимых закономерностей, при том, что результаты могут быть получены строго дедуктивные.

7. Мы и кванторы (5 ч)

База знаний «Классный журнал» (или аналогичная). Факты об отдельных оценках. Вопросы непосредственно к фактам («первому этажу» знаний).

Определение новых понятий в виде правил. Понятие неуспевающего. Процедура поиска ответов на вопросы.

Для получения информации от интеллектуального помощника (в данном случае, компьютера) необходимо четко и однозначно определить понятие, о котором идет речь. Это немаловажный коммуникативный аспект. Создание баз знаний на Прологе вынуждает корректно определять предмет «обсуждения».

Понятие отличника. Трудности, возникающие при составлении правила, вызваны разницей в структуре соответствующих определений (неуспевающего и отличника). Формализация этой разницы приводит к понятию кванторов.

Взаимодополняющий характер и проблема взаимозамены кванторов. Для кванторов можно сформулировать аналоги правил де Моргана.

Наглядное теоретико-множественное представление. Множества отличников и «прочих». Порочный «логический круг» ведет к процедурному «зацикливанию»! Решение проблемы определения отличника.

В качестве заключительного занятия можно провести устный зачет в форме собеседования, заостряя внимание на ключевых моментах курса.

Резерв времени (за год) — 4 ч

ИНВАРИАНТЫ УСПЕШНОГО ЗАВЕРШЕНИЯ КУРСА (7 КЛАСС)

К концу обучения в 7 классе учащиеся должны обладать следующими представлениями, закрепленными в практической деятельности (по разделам).

1. «Модели мышления человека («два этажа»)»

Для понимания сущности информационных процессов разумно «стать на точку зрения» человека, принимающего сообщения, рассмотреть приобретаемое им знание. Моделируя на компьютере представления об «устройстве» мышления человека, мы можем делать выводы о том, насколько глубоко мы вникли в сущность того, как мыслит человек.

В состав знания входят факты. Факт — это суждение о конкретном объекте. Отношение и предикат. Частные и общие суждения.

Отрицание (в формальной логике). Логические связки «и» и «или», их «взаимопереходы» (правила де Моргана). Сложные суждения.

Кванторы существования и всеобщности, их «взаимопереходы» (аналоги правил де Моргана для кванторов).

Понятие рассуждения. Правила, наряду с фактами, входят в состав знания. Процесс рассуждения как последовательное применение правил к имеющимся фактам с целью ответа на фактический вопрос.

2. «Пролог-практикум»

Запись отношений на Прологе. Факты и вопросы к базам знаний. Процедура поиска ответа на вопрос к базе знаний, состоящей из фактов.

Запись правил на Прологе. Процедура поиска ответа к базе знаний, содержащей правила.

Рекурсивное определение отношений. Запись рекурсивных правил на Прологе. Рекурсивный поиск ответа на вопрос.

Общий принцип поиска на вопрос — исчерпывающий перебор всех возможных вариантов.

3. «Творчество с точки зрения взаимосвязи индукции и дедукции»

Индуктивный (от частного к общему) и дедуктивный (от общего к частному) способы рассуждений.

Математика — «царство дедукции». Теоремы; аксиомы и доказательство теорем. Прямая, обратная, противоположная и противоположная обратной теоремы; их истинность и ложность. Логические ошибки в рассуждениях, связанные с подменой одной теоремы другой.

Процесс решения задачи. Чем больше объем имеющегося знания (фактов и правил) тем в меньшей степени это дедуктивный и тем в большей степени творческий процесс.

Индуктивный способ рассуждения. Проблема получения достоверных выводов при обобщающих рассуждениях.

Принцип (метод) математической индукции. Базис индукции (статический компонент) и индуктивный переход (динамический компонент).

Индукция и дедукция — два «крыла», два взаимодополняющих и взаимопереходящих способа рассуждения при решении реальной практической задачи. Пожалуй, лишь индукция может быть отнесена к творческому методу. Ведь даже творчество в решении математических задач заключается в индуктивном (уменьшающем пространство перебора вариантов) поиске нужных «правил».

Продолжение следует

Литература

7 класс

1. *Я. А. Вагрaменко, И. Н. Антипов, Э. И. Кузнецов и др.* «Электронно-вычислительная техника. Учебное пособие для 7—8 классов, под ред. Я. А. Вагрaменко, «Просвещение», 1988.

2. *А. А. Ивин* «Искусство правильно мыслить», М., «Просвещение», 1986.

3. *В. А. Каймин, С. Г. Григорьев, Н. Д. Угринович, А. Г. Щеголев* «Элементы логики и начала языка ПРОЛОГ», журнал «Информатика и образование», № 4, 1989.

4. *А. Г. Щеголев* «Концептуальные основания преподавания предмета «Информатика и моделирование процессов в природе и обществе» (модель «Экология и диалектика»)», в сб. «Педагогический эксперимент «Экология и диалектика», методические рекомендации», выпуск 4, М., МГИУУ, 1990.

5. *Дж. Доорс, А. Р. Рейблейн, С. Вадера* «Пролог — язык программирования будущего», М., «Финансы и статистика», 1990.

6. *С. Г. Григорьев* «Работа системы ПРОЛОГ-Д», журнал «Информатика и образование», № 4, 1990.

7. *С. Г. Григорьев* «Программирование на ПРОЛОГе-Д», журнал «Информатика и образование», № 5, 1990.

А. КУШНИРЕНКО, М. ЭПИКТЕТОВ

КуМир — новое семейство учебных программ

В 1985 г. академик А. П. Ершов с группой соавторов подготовил школьный учебник «Информатика-9». В этом учебнике была введена алголоподобная нотация для записи алгоритмов — так называемый алгоритмический язык. Предполагалось, что алгоритмы, записанные в этой нотации, будут перекодироваться на производственные языки программирования: Бейсик, Паскаль и другие. Летом 1985 г. на механико-математическом факультете МГУ была реализована система программирования на этом языке — Е-практикум (Е — в честь А. П. Ершова). После появления Е-практикума появилось имя «язык Е». Этот язык дорабатывался в учебнике «Информатика-10», а затем в других учебниках информатики [1]. Попутно дорабатывалось и программное обеспечение: Е-практикум был реализован на «Ямахе», «Корвете» и УКНЦ.

Е-практикум — это так называемый редактор-компилятор, система с очень удобным интерфейсом: все синтаксические ошибки мгновенно диагностируются на полях программы, в любой момент программу можно безо всякой задержки начать выполнять во встроенном отладчике. Опыт использования Е-практикума (как на мехмате МГУ, так и во многих других организациях) показал высокую эффективность подобной системы при обучении основам информатики. Однако более широкому распространению Е-практикума мешали слишком строгие ограничения, накладываемые на объем программы, и замкнутость системы. Преодолеть эти недостатки и призвана новая система «КуМир» (название происходит от слов «Комплект Учебных Миров»).

1. Язык КуМир

Разработка языка «КуМир» преследовала цель создать простой язык для начального курса информатики, отвечающий современной технологии программирования и допускающий производственное использование. За основу был взят школьный алгоритмический язык, описанный в учебнике [1]. Язык был дополнен некоторыми возможностями, превращающими его из учебного в производственный. В языке есть:

типы цел, вещ, сим, лит (строки переменной длины), лог;

традиционный набор операций над данными этих типов (включая операции над строками и стандартный набор математических функций);

массивы (таб) указанных типов;

структурные управляющие конструкции циклов, ветвления и др., неструктурный оператор выход;

процедуры с параметрами (делящимися на аргументы и результаты) и локальными переменными, рекурсивные вызовы процедур с параметрами;

операторы последовательного ввода/вывода для работы с файлами, клавиатурой и экраном, графическая библиотека;

возможность написания алгоритмов в машинных кодах для непосредственной работы с памятью и портами (конструкция малг).

Перечисленные возможности перекрывают практически все, что есть в современных структурных Бейсиках, скажем в True Basic или Quick Basic.

Из методических соображений язык включает некоторые не общепринятые мелочи:

допустимость имен, состоящих из нескольких слов, и возможность внесения отрицания внутрь таких имен (например, отрицание утверждения «клетка закрашена» можно записать как «клетка не закрашена»);

возможность опускания пустых скобок при вызове функций без параметров;

запись условий типа «a<b и b<c» в форме «a<b<c».

Язык КуМир открыт — подключение внешних исполнителей обогащает язык новыми возможностями: от управления базами данных и работы с геометрическими объектами до расширения множества допустимых числовых типов (при этом язык позволит смешивать в выражениях новые типы с уже существующими числовыми типами).

И наконец, от структурных Бейсиков КуМир выгодно отличается элементами объектно-ориентированного подхода. В частности, в языке КуМир есть конструкция «исполнитель» (исп).

Современная технология программирования учит разбивать программу не только на подпрограммы, но и на более крупные единицы: наборы подпрограмм, работающих над общими данными. В разных языках программирования такие единицы называются по-разному: «module» (Модула-2), «package» (Ада), «unit» (Turbo Pascal 5.5), «file» (Си). В КуМире такая единица называется «исполнитель». Понятие исполнителя чрезвычайно важно в практической работе и, по нашему мнению, должно быть введено на возможно более ранних стадиях обучения (сначала в облике внешних исполнителей, а затем уже как конструкция языка).

Опыт использования КуМира в преподавании и для разработки учебного программного обеспечения показал, что язык прост в изучении и вместе с тем достаточно мощен для расширения широкого класса производственных задач.

2. Система программирования КуМир

Подобно Е-практикуму, КуМир является интегрированной системой, включающей текстовый редактор, инкрементальный компилятор с нулевым временем ответа, а также простой и удобный отладчик. Хорошее название для системы такого рода — «редактор-компилятор»: пока вы вводите вашу программу, компилятор ее обрабатывает, и в любой момент программа готова к выполнению без малейшей задержки.

Из систем, основанных на аналогичных принципах, упомянем Корнельский программный синтезатор [2] (первая система такого типа), а также ФортранМир [3] и ЭСКОРТ [4].

Мы стремились сделать систему КуМир максимально удобной для новичка. Наш опыт показывает, что в этом случае она окажется удобной и для профессионального использования. Нужно только, чтобы система была инструментом и не навязывала пользователю порядок или стиль выполнения каких-либо действий.

До реализации КуМира мы сформулировали восемь требований к современной системе программирования. КуМир практически полностью удовлетворяет этим требованиям.

1. Для достижения максимальной прозрачности пользовательского интерфейса система должна быть как можно более близка к обычному текстовому редактору.

2. Все статически обнаружимые ошибки должны диагностироваться немедленно во время редактирования программы.

3. «Области влияния» пользователя и системы должны быть четко разделены. Пользователь может изменять текст программы в окне без каких-либо ограничений. Единственное, что позволено делать системе,— это выводить замечания в окно сообщений. Пользователь может реагировать на эти замечания немедленно, а может не обращать на них никакого внимания (и, таким образом, переходить от одного корректного состояния программы к другому через некорректные).

4. Система не имеет права мешать пользователю вводить текст программы или исправлять введенный текст. Например, система не имеет права заменять «a:=σ++1.0» на «a:=b+0.1E1».

5. Ошибки должны диагностироваться как можно детальнее и по возможности немедленно, без каких-либо дополнительных условий со стороны пользователя. Сообщения должны появляться во всех местах, связанных с ошибкой, и исчезать сразу после исправления ошибки.

6. Диагностика и поведение системы должны зависеть только от текущего состояния текста программы (а не от истории ее редактирования).

7. Распространенные ошибки при выполнении (такие как использование неинициализированных переменных или выход индексов за границы массива) должны фиксироваться немедленно; сообщения об этих ошибках должны наглядно привязываться к соответствующему месту программы.

8. Отладчик должен поддерживать автоматическую визуализацию присваиваемых значений и результатов проверяемых условий.

Подобно Е-практикуму и Фортран Миру, КуМир использует метафору «полей» для привязки сообщений системы к тексту программы. Экран поделен на два окна: левое содержит текст программы, правое — «поля программы» (используется для выдачи сообщений компилятора и отладчика). Эти сообщения появляются на полях той строки, к которой относятся. При отладочном выполнении программы результаты вычисления выражений также выводятся на поля. Это позволяет новичку писать и отлаживать не-

большие программы, обходясь без команд ввода/вывода.

Исполнители в системе КуМир

В учебнике [1] постоянно используются учебные исполнители Робот и Чертежник. В КуМире эти исполнители могут быть либо подключены как внешние, либо реализованы на языке КуМир. В любом из этих случаев можно воспользоваться общим механизмом, выделяющим на экране окна для программы и исполнителей.

Разумеется, к КуМиру таким же образом можно подключить и другие учебные исполнители, выбрав их из поставляемого с КуМиром набора (Вездеход, Двуног, Строитель, Куберия и др.) или реализовав самостоятельно.

Внешние исполнители пишутся на языке Си и компилируются отдельно от КуМира (например, компилятором TurboC фирмы Borland). При запуске КуМира указываются имена файлов с исполнителями, которые требуется подключить. Кроме того, при запуске можно указать несколько файлов с текстами на языке КуМир, из которых должна быть собрана программа (что позволяет подключать библиотеки вспомогательных алгоритмов или исполнители, реализованные на языке КуМир).

Например, команда >KUMIR A1.E+ +ROBOT.ISP+ROBLIB.E запускает работу КуМира с программой из файла A1.E, подключает к КуМиру внешний исполнитель ROBOT из файла ROBOT.ISP и библиотеку вспомогательных алгоритмов из файла ROBLIB.E, а команда >KUMIR A2.E+ +SYS.ISP+GT.ISP+ROBOT.E подключает сразу трех исполнителей из файлов SYS.ISP, GT.ISP и ROBOT.E (первые два являются внешними, а третий написан на языке КуМир).

С точки зрения языка КуМир, подключение исполнителя делает возможным вызов команд этого исполнителя. При этом система КуМир будет контролировать правильность типов аргументов и результатов этих команд (типы при этом могут быть стандартными или определенными в подключаемом внешнем исполнителе).

Система КуМир имеет довольно высокие эксплуатационные характеристики. Максимальный объем программы на языке КуМир (в модели MEDIUM, т. е. при использовании коротких адресов) — около 1000 строк (система занимает при этом около 200K оперативной памяти). Переход к модели LARGE незначительно снижает эффективность работы, но увеличивает максимальный объем программы в несколько раз. В режиме с полным контролем (т. е. с проверкой выхода индексов за границы использования неопределенных значений и т. д.) КуМир выполняет программы в среднем в два раза быстрее, чем интерпретатор Microsoft Basic (в котором никакого контроля нет). Производственный режим выполнения (в котором выключены все проверки, а программа переведена в машинные коды) позволяет повысить скорость выполнения в 3—50 раз.

Отладчик КуМира по функциональным возможностям и раскладке команд по клавишам в основном совпадает с отладчиком Turbo Debugger фирмы Borland. Редактор КуМира позволяет копировать символы, строки и конструкции, восстанавливать предыдущие состояния текста (откатка) и по раскладке клавиатуры совпадает с редактором текстов МикроМир.

КуМир работает на всех IBM PC-совместимых машинах с объемом оперативной памяти от 512K и видеоадаптерами MDA, CGA, Hercules, EGA, MCGA или VGA. Для работы достаточно дисковода на 360K; наличие электронного диска или винчестера только ускоряет загрузку КуМира (в процессе работы системы все необходимое находится в оперативной памяти).

КуМир позволяет настроить раскладку клавиатуры, кодировку русских букв на экране и в файле, а также некоторые другие параметры. Внешние исполнители общаются с оборудованием через КуМир, поэтому в отдельной настройке не нуждаются.

Версия системы КуМир на УКНЦ поддерживает работу в классе с одним гибким диском на машине учителя и имеет незначительные ограничения по сравнению с версией для IBM PC.

Разработка учебного программного обеспечения

Общая схема создания учебного «software» с использованием системы КуМир следующая. Критические для эффективности части программы оформляются как внешние исполнители и пишутся на языке Си. Остальные компоненты пишутся на языке КуМир. Таким образом, при программировании общей структуры программы и организации интерфейса используются преимущества редактора-компилятора КуМир (мгновенная диагностика, малое время оборота из-за отсутствия этапа компиляции, отладочные возможности). Эти части системы в наибольшей степени подвержены изменениям в процессе разработки и опытной эксплуатации системы.

Если система имеет фрагменты, отвечающие различным темам изучаемого материа-

ла, то эти фрагменты интегрируются в единое целое при помощи системы КуМир-Гипертекст. Для подачи материала и простейшего тренинга может использоваться универсальная демонстрирующая программа ДемоМир.

КуМир-Гипертекст

Гипертекстовая система сама по себе не является чем-то необычным. По принципу гипертекста устроены многие электронные справочники (например, Tech Help! фирмы Flambeaux Software), встроенные подсказки во многих системах. Однако в системе КуМир-Гипертекст сочетание текста с возможностью программировать поведение системы на некотором простом языке (и к тому же в удобной среде редактора-компилятора) дает принципиально новое качество.

Как и положено в гипертекстовой системе, КуМир-Гипертекст позволяет организовать текст в форме некоторого числа страниц, соединенных ссылками (ссылки при этом могут образовывать произвольный граф). Но, в отличие от электронных справочников, выбор некоторых разделов может приводить не к переходу на другую страницу, а к загрузке (и, может быть, выполнению) некоторого набора внешних исполнителей и программ на языке КуМир. Несмотря на свою простоту, этот механизм позволяет делать достаточно много. В частности, гипертекст может содержать нижеследующее:

1. **Демонстрации, подготовленные при помощи системы ДемоМир.** Эта система позволяет протоколировать клавиатурный ввод (если надо, то с временными задержками) с добавлением поясняющего текста. Кроме того, демонстрация, подготовленная в ДемоМире, может включать элементы тренинга — в ней может быть предусмотрено нажатие пользователем некоторой последовательности клавиш.

2. **Демонстрирующие и/или контролирующие программы, написанные на языке КуМир.** Разумеется, такие программы могут использовать все возможности механизма подключения исполнителей и библиотек.

3. **Практикумы.** В этом случае КуМир загружает две программы (и, если надо, произвольный набор внешних исполнителей, а также исполнителей и библиотек на языке КуМир). Первая программа доступна для редактирования и обычно содержит шаблон алгоритма, который ученик должен написать или исправить. Вторая программа содержит контролирующий модуль (написанный также на языке КуМир!), который вызывается при нажатии учеником клавиш «проверить решение» или «дать подсказку» (а также в некоторых других случаях, например, при входе в задание и выходе из него). Контролирующий модуль может вызвать написанный учеником алгоритм с разными входными данными и проверить результат его работы, дать необходимые пояснения (зависящие от того, насколько ученик продвинулся в решении задачи) и т. д.

Заметим, что описанные выше типы разделов не встроены жестко в систему, а задаются сочетанием программ, исполнителей и ключей, управляющих режимом выполнения. Это позволяет комбинировать их произвольным образом, добиваясь требуемой гибкости.

Для сбора результатов работы ученика используется механизм «журнала». С точки зрения контролирующей программы журнал является просто массивом строк (лит таб). При работе с заданием элементы этого массива выводятся в заранее определенные места гипертекста. Строка журнала может содержать оценку действий ученика (включая затраченное на выполнение задания время), рекомендацию по дальнейшей работе или может использоваться для связи между отдельными компонентами курса.

Прикладные пакеты программ

Достаточно высокие эксплуатационные характеристики и механизм подключения внешних исполнителей, имеющийся в системе КуМир, позволяют использовать систему не только для обучения, но и в качестве инструментальной при написании прикладных программ. Ниже перечислены некоторые внешние исполнители, которые можно использовать при создании таких программ.

ГраТекс и BGI. Два графических пакета примерно одного уровня. Первый был разработан на мехмате МГУ специально для системы КуМир и содержит оригинальные высокоэффективные алгоритмы машинной графики. Основное достоинство второго пакета — совместимость по интерфейсу с графическим интерфейсом BGI фирмы Borland; использование этого пакета упрощает переход от системы КуМир к профессиональным средам Turbo Pascal и Turbo C.

Куберия. Очень простой по интерфейсу исполнитель, позволяющий создавать из кубиков реалистические трехмерные картинки.

МикроМир. Подключение профессионального текстового редактора в качестве внешнего исполнителя позволяет, с одной стороны, писать сложные программы обработки текстов, а с другой, использовать систему КуМир для расширения набора команд ре-

дактора (например, для внесения элементов электронных таблиц).

Функции и Графики. Пакет высокого уровня, поддерживающий все разделы курса алгебры для 7—9 классов средней школы, в которых идет речь о функциях и их графиках: понятие функции, область определения, график, линейная квадратичная, степенная, дробно-линейная функция, модули от функций.

КуМир-Гео и КуМир-Стерео. Два пакета для визуализации геометрических разделов на плоскости и в пространстве соответственно. При подключении одного из этих пакетов к языку добавляется новый тип «фигура». Пользователь может заводить переменные и массивы данных этого типа, выполнять различные геометрические операции; все производимые действия автоматически отображаются на экране.

КуМир-СУБД. Подключение к КуМиру внешнего исполнителя для работы с базами данных (написанного на основе системы Paradox Engine) превращает КуМир в систему управления базами данных, сравнимую по возможности с промышленными СУБД, но значительно превосходящую их по удобству программирования и гибкости.

Возможность произвольно комбинировать используемые внешние пакеты позволяет создавать на простом и удобном языке такие сложные системы, как, например, текстовый редактор с элементами деловой графики или базу данных геометрических объектов.

Курсы с использованием КуМира

В заключение перечислим курсы, уже созданные или еще разрабатываемые на базе системы КуМир:

информатика 10-11;
информатика 7-8;
информатика 5-6;
занимательная информатика;
программирование для старшеклассников;
практикум по программированию на языке Си;
Функции и Графики (по книге И. М. Гельфанда, Е. Г. Глаголевой и Э. Э. Шноля «Функции и графики», серия «Библиотека математического кружка», вып. 2, М., Наука, 1973);
предгеометрия;
задачи на построение в планиметрии (с использованием пакета КуМир-Гео);
сечения многогранников (КуМир-Стерео);
начала анализа (с использованием пакета Функции и Графики);
основы Финформатики (финансовой информатики).

Литература

1. Основы информатики и вычислительной техники: Учеб. для сред. учеб. заведений. / А. Г. Кушниренко, Г. В. Лебедев, Р. А. Сворень.— М.: Просвещение, 1990.

2. T. Teitelbaum, T. Reps. The Cornell Program Synthesizer: A systax—directed programming environment // Communikations of the ACM. September 1981, v. 24, N 9.

3. *Варсонофьев Д. В., Дымченко А. Г.* Ф-практикум — учебно-инструментальная среда для разработки программы на Фортране. // Седьмой семинар «Проблемы информатики и ее применение в управлении, обучении и научных исследованиях». Аннотации докладов. София, 1988.

4. *Галатенко В. А., Митропольский А. Ю., Прилипко Г. А., Христов П. В.* Основные компоненты системы ЭСКОРТ // «Вопросы кибернетики. Проблемы автоматизации инженерного труда в машиностроении». М.: НСК АН СССР, 1988. С. 25—34.

В. ГИСИН, О. КИРЮШКИНА, А. МОРДКОВИЧ
Московский государственный открытый педагогический институт

Графический калькулятор TI-81 для школы

TI-81 — первый графический калькулятор, разработанный специально для математического образования фирмой Texas Instruments (США). Калькулятор TI-81 используют около 10 % американских школьников. В настоящее время начинается его освоение в школах Европы. Появились первые образцы и в России.

Графический калькулятор TI-81 — одно из наиболее прогрессивных направлений применения новых информационных технологий в преподавании математики. Предоставляя в распоряжение учителя и ученика специфические возможности по быстрому выполнению значительных объемов рутинной работы, системы инструментального типа (к числу которых относится и TI-81) позволяют добиться нового качества в математическом образовании школьников.

Перечислим возможности, которые предоставляют пользователю инструментальные математические системы.

1) Вычисления значений математических выражений.

2) Программирование.

3) Вычисления со сложными данными (статистические расчеты, операции с векторами и матрицами, численное дифференцирование и интегрирование и т. п.).

4) Построение графиков.

5) Графическое представление статистических данных.

6) Символьные преобразования.

Универсальные системы, обладающие всеми перечисленными возможностями, требуют значительных ресурсов вычислительной техники и реализуются на компьютерах.

К числу простейших инструментальных математических систем можно отнести обычные калькуляторы, которые позволяют производить вычисления с элементарными функциями. На более мощных калькуляторах первого поколения можно, кроме того, программировать и выполнять статистические расчеты.

Калькулятор TI-81, относящийся к калькуляторам второго поколения, снабжен развитой системой графического отображения информации и предоставляет пользователю богатый набор операций, реализующих указанные выше возможности.

Приведем краткое описание устройства и технические характеристики калькулятора TI-81.

Калькулятор TI-81 работает от 4 батареек типа AAA. Блок питания находится на тыльной стороне корпуса. На лицевой стороне корпуса расположены клавиатура и дисплей. Ввод информации в калькулятор производится с клавиатуры, вывод информации осуществляется на дисплей.

В соответствии с функциональным назначением, клавиши клавиатуры условно разделяются на 4 зоны: «Графика», «Редактирование», «Научный калькулятор», «Дополнительные возможности». Клавиши разных групп отличаются по цвету.

Дисплей используется в двух режимах: символьном и графическом. При работе в символьном режиме на дисплее отображается 8 строк, по 16 символов в каждой. При работе в графическом режиме на экране располагается изображение, составленное из черно-белых элементов («точек»); разрешающая способность — 64 точки по вертикали, 96 точек по горизонтали.

При вычислениях в калькуляторе используется представление чисел с 13-ю десятичными знаками. На дисплей выводятся значения, содержащие 10 десятичных знаков (по желанию пользователя количество выводимых знаков может быть уменьшено — для этого нужно заранее зафиксировать соответствующую команду при выборе режима работы).

Калькулятор TI-81 обладает оперативной памятью объемом 2400 байтов, в которой могут размещаться данные и программы. Одновременно в памяти калькулятора может находиться до 37 программ. Содержимое оперативной памяти и текущие значения переменных сохраняются при выключении калькулятора за счет подпитки памяти от батареек. При смене батареек или перезагрузке калькулятора содержимое оперативной памяти стирается и устанавливаются исходные стандартные значения всех переменных.

Подробная инструкция по работе с калькулятором TI-81 (на английском языке) представляет собой 150-страничное пособие, но наш опыт обучения студентов педагогического института показал, что практически за 2—3 недели калькулятор осваивается студентом как пользователем, а еще за такой же период времени и как программистом.

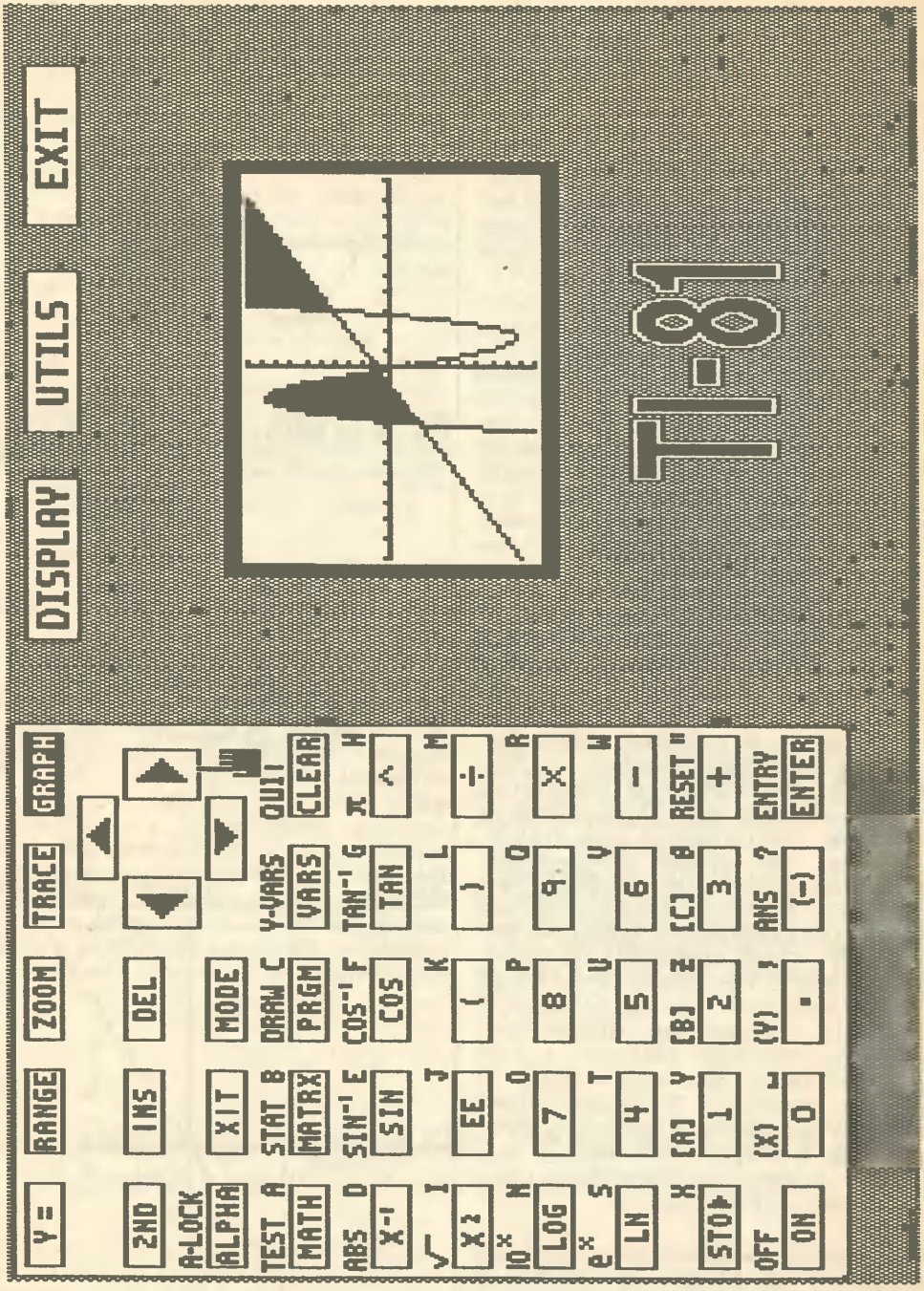
Чтобы читатель имел некоторое представление о калькуляторе TI-81, опишем его клавиатуру (рис. 1).

Большинство клавиш калькулятора может использоваться в трех режимах: основном, режиме второй функции, алфавитном.

Основная функция клавиши указана непосредственно на ней самой; вторая функция — слева сверху от клавиши; буква, связанная с клавишей в символьном режиме,— справа сверху от клавиши. Режим второй функции устанавливается нажатием клавиши 2ND, алфавитный режим — нажатием клавиши ALPHA. Действие установленного режима отменяется после первого нажатия любой из указанных двух клавиш вслед за установкой.

Рядом с клавишами 2ND и ALPHA находятся клавиши INS и DEL, которые используются для редактирования при вводе информации, а также клавиши со стрелками для управления перемещением курсора. Здесь же находится клавиша x/t, используемая для записи переменной X (или T — в случае, если строятся графики параметрически заданных функций), и клавиша MODE — меню. Используя MODE — меню, можно задать фиксированное количество десятичных знаков в получаемых числах, выбрать радианную или градусную меру углов, обычный Y(x) или параметрический X(T), Y(T) способ задания функции, динамический или статический способ выведения графика на дисплей и т. д. В левом нижнем углу калькулятора расположена клавиша ON [OFF], которая используется для включения (выключения) калькулятора. Справа внизу клавиша «ввода» ENTER.

Непосредственно под дисплеем расположены 5 графических клавиш:



1) Y — используется для аналитической записи функций, графики которых нужно построить;

2) RANGE — с помощью этой клавиши можно установить размеры «окна обзора» графика;

3) ZOOM — это режим «лупы». Используя команды из ZOOM-меню, можно увеличить (рассмотреть «под лупой») часть изображения;

4) TRACE — эта клавиша заставляет курсор двигаться вдоль графика, при этом на экран выводятся координаты курсора;

5) GRAPH — команда для построения графика указанной функции.

В четвертом ряду сверху располагаются

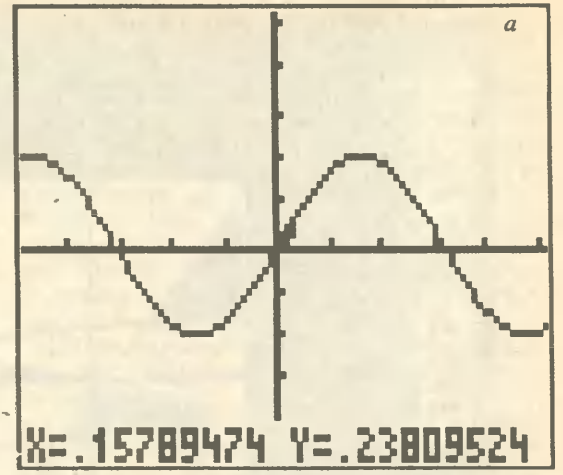
клавиши «дополнительных возможностей». С их помощью можно: производить вычисления с матрицами, выполнять статистические расчеты, обращаться к режиму программирования, использовать интерактивную графику, использовать некоторые дополнительные математические функции (например, находить численное значение производной) и т. д. Еще ниже расположены клавиши, с помощью которых можно производить арифметические операции и вычислять значения элементарных функций.

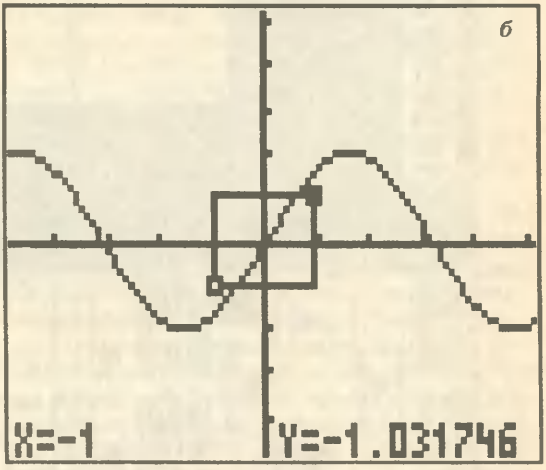
Калькулятор TI-81 вполне приспособлен для изучения алгебры и начал анализа в курсе математики средней школы, для внедрения в учебный процесс при изучении различных курсов высшей математики, для постановки математических экспериментов, для инженерных вычислений. Калькулятор TI-81 по своему потенциалу способен играть роль специализированного компьютера.

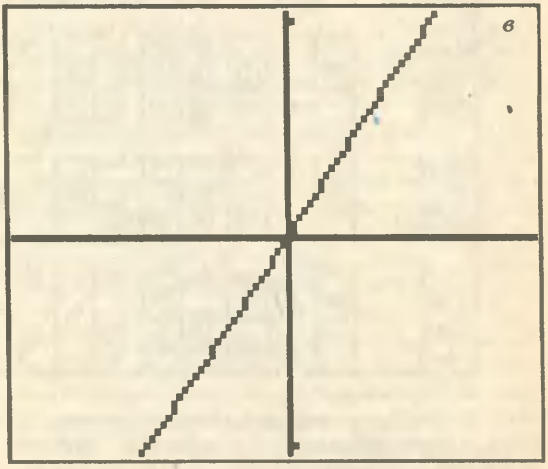
Назовем характерные черты учебной работы с использованием инструментальной системы такого типа.

1. С ученика снимаются заботы, связанные с проведением целого ряда стереотипных стандартных операций. В частности, если говорить о графических возможностях, то, воспользовавшись ими, можно получать поточечно построенные графики функций, что переносит акцент в изучении графиков — вместо цели они становятся средством изучения, что создает в обучении благоприятный психологический фон. При традиционном подходе доминирующая цель — снабдить ученика некоторыми стандартными приемами построения эскиза графика. Речь обычно идет о выявлении набора (довольно ограниченного) качественных характеристик: область определения, монотонность, периодичность, четность. При этом остается в тени прикладная направленность задачи построения графика. Применение графического калькулятора позволяет включить в круг изучения реальные числовые значения с приемлемой точностью. Благодаря этому появляется возможность не только продекларировать, но и показать школьникам возможности практического (на учебных задачах) применения графиков.

2. Изучение основ анализа связано с необходимостью овладеть рядом абстрактных, а потому достаточно трудных понятий. Трудность усугубляется недостаточным объемом эмпирического материала. Последнее вполне объяснимо, если учесть, что построение даже сравнительно простых графиков отнимает у школьников довольно много времени. Применение TI-81 позволяет расширить и сделать качественно более разнообразным набор графиков, с которыми школьники могут практически работать. Это создает хорошую осно-







2

ву для гармонического сочетания образных и формально-логических методов при изучении понятий в курсе начал анализа.

3. Использование калькуляторов позволяет организовать на уроке в обычном классе лабораторные работы, проводить ученические мини-исследования. В методическом плане благодаря этому восстанавливается звено «от накопления фактов — к их теоретическому осмыслению», которое в значительной степени потеряно при традиционном школьном преподавании и очень важно в овладении основами науки и для развития научного мышления. Следует иметь в виду, что при работе с графическим калькулятором резко активизируется самостоятельная учебная деятельность ученика, снижается управляющее воздействие учителя на процесс овладения знаниями. Для компенсации последнего явления нужно предусматривать соответствующие формы организации целенаправленной самостоятельной работы учащихся на уроке.

4. Имеется еще один важный аспект уже педагогического плана, связанный с предыдущим замечанием. Выполнение лабораторных работ требует от учащихся активной и разнообразной деятельности. Причем многие виды этой деятельности (особенно на этапе накопления фактов) доступны для учеников со значительными пробелами в знаниях. Разница в подготовленности учащихся и их способностях сказывается, главным образом, на этапе теоретического осмысления. Каждый учащийся независимо от предварительной подготовки в состоянии добиться успеха (мера этого успеха будет, конечно, разной у разных учащихся). Благодаря этому у учащихся формируется положительное отношение к изучаемому материалу, возникают дополнительная мотивизация, интерес к учебе.

Одним из примеров использования калькулятора TI-81 в школьном курсе начал анализа может служить следующий подход к понятию производной.

Операции изменения окна обзора (ZOOM) дают прекрасную иллюстрацию понятию дифференцирования. При достаточном увеличении гладкие кривые выглядят на чертеже как прямые, т. е. локально (в достаточно малой окрестности рассматриваемой точки) гладкая функция не отличается от прямой (см. рис. 2).

Здесь частично раскрывается основная идея математического анализа — изучать локальное поведение функций и по нему делать выводы об их глобальных свойствах. В режиме «лупы» гладкие кривые выглядят как прямые, основная характеристика прямой на координатной плоскости — ее угловой коэффициент, т. е. производная. Отсюда два вывода: первый, производная — это локальная характеристика функций и второй, основная идея дифференцирования — идея линеаризации (замены графика функции прямой линией). Подобное рассуждение, при всей его нестрогости, довольно точно отражает суть дела и служит методологическим обоснованием введения довольно сложного и громоздкого по школьным меркам понятия производной.

Калькулятор TI-81 на порядок дешевле любого компьютера, используемого ныне в школах и других учебных заведениях, и комфортнее в использовании (не требует специальных помещений).

Редакция журнала располагает подробной информацией о калькуляторе TI-81, опыте его использования в учебном процессе, возможностях приобретения для образовательных учреждений, других организаций и граждан.

А. ШМЕЛЕВ

Компьютерная поддержка самостоятельной работы по усвоению лексики иностранного языка

Цель данной статьи — рассмотреть указанную проблему в контексте взаимодействия так называемых «новых информационных технологий» с имеющимися в современной когнитивной психологии представлениями о строении и функционировании лексической (словесной) памяти, а также о когнитивном стиле личности. Важнейшие психологические требования к системам компьютерной поддержки «вокабулярного самообучения» проиллюстрированы автором конструктивными решениями, реализованными в собственной разработке — в программных средствах СЕЛАБ и ЛЕКСИТРАН.

Дело в том, что по определенным методологическим причинам внимание теоретиков и практиков-конструкторов подобных систем надолго (фактически на три десятилетия) оказалось прикованным лишь к одному, безусловно, центральному, но не единствен-

ному аспекту проблемы — к проблеме создания алгоритмов оптимизации предъявления учебного материала (словарных заданий) с учетом индивидуальной истории обучения конкретного учащегося. Это проблема «адаптивности» обучающих программ. В качестве примера такого акцента можно указать на классическую работу Ричарда Аткинсона [1]. Из отечественных исследователей наибольшую известность в этой области получили работы Л. А. Растригина [6]. В журнале «Информатика и образование» публиковалась статья Л. А. Растригина [7], включающая ссылку на конкретную программную систему «АССОЛИЯ», основное достоинство которой автор прежде всего усматривает в адаптивности алгоритма обучения на базе модели учащегося.

Такой акцент на проблеме адаптивности явился, по нашему мнению, естественным следствием подхода к обучению как к процессу управления. Кибернетическая парадигма применительно к обучению на долгие годы связала практику использования компьютеров с задачами программированного (то есть адаптивного) обучения (см.: [9, 5, 8]). В этой логике компьютер в обучении прежде всего рассматривался как управляющая машина, а учащийся как объект обучающих воздействий. И до тех пор, пока доступ учащегося к компьютеру был регламентирован администрацией крупных многотерминальных управляющих ЭВМ (и соответствующих учебных классов, как это было в нашей стране в рамках АОС на базе ЕС ЭВМ, или мини-ЭВМ серии СМ), позиции узко-кибернетического взгляда на вопрос казались неколебимыми, несмотря на всю методологическую критику его со стороны психологов.

Но «персональная революция» 80-х гг. [3] и победа принципа свободного рыночного распространения программного обеспечения для персональных компьютеров привели к тому, что «вокабулярные тренажеры», основанные на модели пассивного ученика, являющегося объектом «оптимальных» обучающих воздействий, стали просто пылиться на полках софтвермаркетов в соответствующем разделе «educational software». (Подобное автору статьи доводилось наблюдать и в США, и в Европе; нет сведений у автора и о крупном коммерческом успехе таких отечественных программ, как упомянутая «АССОЛИЯ», или, например, «Thousand», принятой на дистрибуцию в СП «Диалог»).

Итак, очевиден парадоксальный разрыв между явно выраженной потребностью (особенно у наших отечественных массовых пользователей персональных ЭВМ) в освоении лексики иностранного языка и низкой популярностью языковых обучающих программ. Почему же пользователи не хотят обучаться «оптимально»? То, что мы наблюдаем,— это значительно большая активность в использовании резидентных словарей (для читателей, не знакомых с этой технологией, поясним, что резидентный словарь позволяет получать перевод непонятного слова на экране монитора в процессе просмотра иностранного текста — достаточно подвести курсор под это слово и нажать обусловленную комбинацию клавиш). Именно резидентные англо-русские словари стали стихийно распространяться, превратившись в предмет натурального обмена.

Самое поверхностное объяснение более высокой популярности резидентных словарей сводится к ссылке на «леность» пользователей: зачем, мол, учить тысячи иностранных слов, когда в любой момент можно подвести курсор к незнакомому слову и тут же на экране вы имеете необходимый перевод. Два-три раза проделал это со словом — и вот оно уже само запомнилось. Ну а если через неделю это слово забылось, тоже не беда — словарь всегда под рукой.

Такой подход к делу следует признать вполне самодостаточным прежде всего для «хакеров» — «заядлых пользователей»: ведь всю информацию они черпают в режиме считывания ее с экрана монитора, и резидентный словарь (имей он только достаточный объем) всегда выручит. Другое дело пользователи, нередко отрывающиеся от компьютера для чтения книг — им важно закрепить значения иностранных слов, чтобы пореже лазить в громоздкий и неудобный словарь.

Более глубокое психологическое объяснение такого предпочтения может дать нам толчок для некоторых новых конструктивных идей. В экспериментальных исследованиях психологи установили, что комфортнее в условиях традиционного компьютерного обучения по типу drill-and-practice (оптимизированной «зубрежки») себя чувствуют люди с так называемым «пассивно-импульсивным» когнитивным стилем, по сравнению с «активно-рефлексивными» субъектами [2]. Последним навязываемая компьютером «стимуляция» (учебный материал) мешает производить необходимую внутреннюю работу по воссозданию собственной целостной системной концепции (образно-концептуальной модели) изучаемого материала.

Наши собственные исследования, проведенные совместно с Ю. В. Фомичевой [10], показали, что заядлые пользователи (в частности, заядлые игроки в компьютерные игры) обладают чертами субъектов с активным

когнитивным стилем: у них более высокий уровень интернальности в «локусе контроля» (приписывают ответственность за успехи и неудачи себе самим, а не внешним обстоятельствам), более высокий уровень социальной смелости и одновременно аналитичности мышления. Следовательно, нет ничего удивительного в том, что любители и знатоки компьютеров предпочитают именно такие виды программного обеспечения, которые оставляют за ними активную позицию «хозяина положения», то есть субъекта, скорее заставляющего компьютер помогать себе, а не «пациента», принужденного пассивно отвечать на вопросы и задания, которые «ставит» компьютер. Ведь сеанс тренинга в работе с «дриллером» (тренажером) эффективен только тогда, когда вы имеете терпение и время высидеть его до конца, еще раз проникнувшись пафосом древнего лозунга «повторение — мать учения». Но активному пользователю постоянно хочется иметь возможность самому двигаться по материалу, самому планировать маршрут. Это одно психологическое обстоятельство. Назовем его **фактором «активности»**.

Второй важнейший психологический фактор состоит в стремлении самодеятельных пользователей решать собственные осмысленные задачи. И здесь резидентный словарь оказывается просто находкой — он позволяет совместить чтение интересующих текстов с подучиванием словарного запаса. Причем такое подучивание в процессе чтения значимых текстов не случайно оказывается более эффективным, чем часы, проведенные за зубрежкой. Дело в механизмах и закономерностях человеческой памяти, описанных еще в классических работах советских психологов П. И. Зинченко и А. А. Смирнова: тот материал эффективно запоминается, который входит каким-то значимым элементом в функциональную структуру внутренне (а не внешне!) мотивированной деятельности человека (в качестве условия, цели или самого мотива).

Объяснить читателям эту закономерность проще всего таким образом: слова, которые вы заучили в процессе специальной работы по заучиванию, имеют тенденцию вспоминаться, когда вы опять занимаетесь именно заучиванием, но когда вы занимаетесь чтением осмысленного текста (и тем более — его порождением, и тем более — на фоне коммуникативного стресса в ситуации реального общения), они вовсе не «всплывают». Итак, второй фактор — это **фактор «значимого контекста»** (имеется в виду прежде всего экстралингвистический контекст — обстоятельства реальной жизнедеятельности учащегося, лежащие за пределами сферы самого языка). Использование принципов «активности» и «значимого контекста» — краеугольные камни «коммуникативного метода интенсивного группового обучения языку», который в нашей педагогике языкового обучения представлен прежде всего Г. А. Китайгородской и ее последователями.

Низкая эффективность адаптивных вокабулярных тренажеров связана и с отсутствием в большинстве из них структурации учебного материала по семантическому принципу (в лучшем случае имеется группировка материала по грамматическим классам — глаголы к глаголам, предлоги к предлогам и т. п.), а также с отсутствием прямых заданий, направленных на активизацию внутренних умственных действий учащегося по семантической категоризации учебного материала.

Что мы понимаем, собственно говоря, под «зубрежкой»? Установление прямых ассоциативных связей между двумя словоформами (иностранным и родным словом), минуя семантическую репрезентацию (то есть активизацию в сознании понятийных, образных, эмоциональных и других аспектов значения слова). Как в современных обучающих системах пытаются преодолеть этот недостаток? Прежде всего — созданием эффективной образно-графической поддержки учебного материала. Хотя даже на обычных винчестер-дисках (а во многих наших учебных классах пока и они не доступны — идет работа на компьютерах без встроенных дисководов) разработка нескольких сотен простеньких пиктограмм (отсканированных и раскрашенных картинок) оборачивается значительными затратами памяти (счет очень быстро начинает идти на мегабайты!). И все же, в преддверии массового распространения лазерных дисководов с их гигабайтами памяти разработчики уже сейчас направляют свои усилия на обеспечение красочной иллюстративности учебного материала. Спору нет (и в строгих научных экспериментах было показано [14]), что обучение знакам нового языка (а для строгости брался псевдоязык из бессмысленных слогов и фраз) идет гораздо быстрее при наличии визуальной поддержки. Назовем «визуальную поддержку» третьим фактором психологической эффективности вокабулярных тренажеров. В общем виде этот фактор, конечно, должен быть назван «сенсорной поддержкой» (сюда с известным огрублением можно отнести и «звуковой» образ слова, и иллюстративные шумовые эффекты и т. п.,— все то, что так ценится в новомодных обучающих средах «мультимедиа»).

К сожалению, немногими пока осознано, что информационно-компьютерная революция на сегодняшнем этапе в силу коммерциализации оказывается все меньше «интеллектуальной революцией для масс», но все больше технологическим средством «видеокультурной интеллектуальной контрреволюции». Засилье видеокультуры в коммерческих компьютерных играх (акцент на внешние эффекты, как правило, в ущерб стройности скрытой логико-семантической концепции игры) грозит перекочевать и в сферу «educational software».

По нашему убеждению, визуальная поддержка не может заменить собой необходимость активизации процессов построения категориальной иерархии для лексики иностранного языка. Здесь уместно привести предложенное еще в 60-е годы Тулвингом различение «эпизодической» и «семантической» памяти [4]. Зрительные образы, зрительные ассоциации привязывают слова к определенным «эпизодам», конкретным по месту и времени. При этом затрудняется обобщение поля значения слова. Собственно «семантическая» память организуется в структурах живого индивидуального знания в системе «категориально-иерархических» ассоциативных связей (связей между родовыми понятиями, видовыми понятиями и атрибутами — достаточно напомнить широко известные модели «категориальных деревьев» Коллинза и Квиллиана [4].

Во многих экспериментах показано [13, 11], что выполнение задачи на категоризацию (семантическую классификацию) — является одним из наиболее продуктивных по своему мнемотехническому эффекту видов умственной деятельности. Это происходит потому, что в сознании человека устанавливаются ассоциативные связи наиболее обобщенных отношений категориальной иерархии.

Таким образом, четвертый выделенный нами фактор — «семантическая генерализация» (освоение слова в его максимально обобщенном, генерализованном значении). Подчеркнем, что имеется известное противоречие между действием этого фактора и фактора «значимого контекста». Как и эффективная визуальная поддержка, значимый контекст приводит к известному сужению семантического поля осваиваемого слова. И чем более контекст значим, тем с большой вероятностью освоенное слово может оказаться узко «привязанным» только к данному контексту.

Проиллюстрируем выделенные нами 4 фактора некоторыми конструктивными решениями, которые мы пытались впервые реализовать в наших программах по «вокабулярному обучению» — СЕЛАБ и ЛЕКСИТРАН*.

Игровая программа СЕЛАБ

СЕЛАБ — компьютерная развивающая игра «СЕмантический ЛАБиринт». Задача игрока состоит в том, чтобы найти путь к заданному пункту в некотором лабиринте, узлы которого обозначены словами иностранного языка. В структуре лабиринта реализована семантическая сеть — соседние слова находятся в отношениях определенного семантического родства (синонимии, антонимии, родовидовой иерархии, атрибутивной принадлежности и т. п.). Если игрок правильно понимает смысл слов, то он может быстрее отыскать кратчайший путь и затратить на его прохождение минимальное количество времени. При этом сама по себе учебная задача «выучить» является лишь необходимым условием решения чисто игровой задачи, а заучивание значений слов — как бы «побочный» развивающий результат игровой деятельности.

Как всякая игровая программа, СЕЛАБ провоцирует чисто игровую спортивную мотивацию — ведутся глобальные и локальные (для частных лабиринтов) таблицы рекордов, обеспечиваются дополнительные игровые стимулы и эффекты.

Кроме самого игрового диалога программа снабжена инструментальными сервисными блоками, универсализирующими возможности ее использования. Во-первых, это диалоговый редактор лабиринтов, позволяющий создать лабиринт в несколько десятков узлов для любой области лексики. Единственное (довольно существенное) ограничение — каждый узел можно связать не более чем с тремя соседями одновременно (узел на экране — «комната с тремя дверями»), но уже такая простая структура позволяет моделировать, например, категориальные деревья. Второе инструментальное средство — автоматический генератор лабиринтов произвольной (задаваемой пользователем) глубины для семантической базы данных табличной формы «слова*семантические факторы», в которой слова описаны в виде многокритериальных векторов с компонентами в виде факторных нагрузок (семантических маркеров). В настоящем варианте такая база данных размерностью 1650*15 реализована для русской и английской лексики личностных черт, а 15 факторов были получены нами в рамках ТЕЗАЛ (компьютеризированный тезаурус личностных черт) [12].

В специальных экспериментах, проведенных под нашим руководством Ю. А. Машинцевым, было показано, что скорость

прохождения лабиринтов значимо коррелирует с семантической компетентностью, измеренной с помощью теста когнитивной сложности для данной области лексики [11]. Таким образом, можно быть уверенным, что игра СЕЛАБ повышает семантическую компетентность пользователя даже в рамках категориальных структур родного языка, не говоря уж о полезном эффекте заучивания лексики языка иностранного.

Какие же психолого-педагогические факторы иллюстрирует игра СЕЛАБ? В первую очередь — это факторы «активности» и «семантической генерализации». Самостоятельно и активно выбирая маршрут в «семантическом лабиринте», пользователь, безусловно, более свободен и инициативен, чем в традиционном адаптивном тренажере. Прослеживая семантические связи слов, пользователь непроизвольно усваивает семантические поля в их системной взаимосвязи (принцип «семантической генерализации»).

Безусловно, менее решенными в рамках СЕЛАБа остаются пока проблемы «значимого контекста» и «визуальной поддержки», хотя в последнем случае нет принципиальных ограничений: продуктивная бригада компьютерных художников вполне может «прописать» в комнатах лабиринта яркие иллюстрации для соответствующих понятий.

Словарь-тренажер ЛЕКСИТРАН

ЛЕКСИТРАН — программа, более непосредственно решающая задачи вокабулярного обучения (ЛЕКСической ТРЕНировки). Она включает блок адаптивного тренинга, практически идентичный по своему смыслу тому, что имеется в обычных (профессионально сконструированных) вокабулярных тренажерах.

Адаптивный блок управляет оптимизацией процесса повторения. Реализованный в ЛЕКСИТРАНе адаптивный алгоритм моделирует все параметры универсальной модели адаптивного обучения, описанной в работе Р. Аткинсона [1].

Но как мы уже отмечали, «пафос» разработки словаря-тренажера ЛЕКСИТРАН состоял не в модели оптимального повторения, а в попытке учесть четыре обозначенных выше психологических фактора.

Для обеспечения более активной позиции пользователя в работе с ЛЕКСИТРАНОМ, мы спланировали эту программу прежде всего как инструментальную оболочку, открытую для наполнения материалом, интересующим самого пользователя (в американской классификации софтвера ЛЕКСИТРАН — типичный пример так называемой authoring system). Большинство языковых компьютерных «тьютеров» (программ-наставников) предлагают ученику заранее подготовленный, «зашитый» в них учебный материал. Пользователь ЛЕКСИТРАНА обеспечен возможностью настраивать нужные ему режимы в работе с нужным ему материалом (главное меню снабжено субменю настройки, кроме того механизм повторения или выдачи статистики можно включать и выключать в ходе самого обучающего сериала). Пользователь может осваивать один и тот же список слов в режиме перевода на иностранный и обратного перевода на родной язык; в режиме угадывания (самоконтроля правильности ответа) и строго побуквенного компьютерного контроля (спеллинг-контроль); в режиме контекстной вставки с разными размерами экспонируемого текстового фрагмента (одна строка, три строки, целый экран); в режиме тренинга и тестирования (без обратной связи о правильном ответе); в режиме категоризации и т. п. Таким образом, адаптивный тренинг в ЛЕКСИТРАНЕ дан пользователю лишь как одна из возможных альтернатив в работе со словарным материалом. Но главное — это возможность создания и ведения персонализированной базы данных: ЛЕКСИТРАН снабжает пользователя средствами упорядочивания лексики по файлам и группам файлов (собранных в специально заводимые пользователям директории) любым удобным для него способом.

Важнейший элемент ЛЕКСИТРАНА, реализующий принципы «активности» и «значимого контекста», заключается в его особом функциональном сопряжении со специально созданной для этих целей модификацией редизентного словаря РИСС (разработчик — С. Р. Мизрохи, ИПИАН). Предположим, пользователь читает с экрана присланное ему по электронной почте сообщение какого-то зарубежного корреспондента: достаточно подвести курсор (удобнее мышкой) под незнакомое слово и тут же вспыхивает окно с несколькими вариантами перевода на русский язык. Но кроме обычного резидентного словаря, модификация РИСС+ЛЕКСИТРАН одновременно со справочным обслуживанием реализует подготовку данных в файл-протокол для тренинга — запрошенное слово с его переводом поступает в указанный пользователем (при загрузке РИСС) текущий файл-протокол в формате, принятом в ЛЕКСИТРАНЕ. Тем самым «авторизация» учебного материала для ЛЕКСИТРАНА практически не требует от пользователя никакой дополнительной работы, кроме обычной работы по использованию резидентного словаря! Отдельного

комментария заслуживает иная, противоположная по последовательности запусков комбинация двух сопряженных в одну систему программ ЛЕКСИТРАН+РИСС: когда пользователь работает уже в запущенном для тренинга ЛЕКСИТРАНЕ и никак не вспомнит спеллинг заданного ему слова, он может вызвать резидентный словарь и отыскать нужное слово (воспользовавшись сервисом ускоренного побуквенного поиска, реализованного в РИСС). Найденное таким образом слово «отбрасывается» программой РИСС (по нажатию комбинации SHIFT+ENTER) в буфер ответа программе ЛЕКСИТРАН.

В нынешней версии ЛЕКСИТРАН-2.5 также целенаправленно реализован принцип «семантической генерализации и систематизации» приобретаемых лексических знаний.

Для этого автором (на базе анализа современных тезаурусов и разговорников) была разработана особая сетка из 167 мезокатегорий лексики, объединенных в 13 макрокатегорий. Приведем здесь только макрокатегории, обозначенные терминами английского языка (даются в алфавитном порядке): BUSINESS, CULTURE, HEALTH, HOME, NATURE, PERSON, PLEASURE, SCIENCE, SERVICE, SHOPS, SOCIETY, TALKS, TRAVEL.

Эти макрокатегории использованы как имена субдиректорий в генеральной субдиректории TOPICS в рамках системы ЛЕКСИТРАН.

И дело здесь вовсе не в особой научной или культурной ценности разработанного варианта категориальной иерархии, а в реализации самого подхода. Пользователю предлагается возможность поместить новое встретившееся ему слово (из текущего файла-протокола) в один из 167 файлов системы TOPICS, что дает возможность закреплять у него необходимое умственное действие по семантическому обобщению и четкой локализации значения слова (это реализуется в новом для вокабулярных тренажеров режиме «тематической классификации»).

Таким образом, в различных субдиректориях ЛЕКСИТРАНА смоделированы (как бы экстериоризированные в памяти компьютера) два параллельных вида человеческой памяти — «эпизодическая» (максимально связанная с ситуационным контекстом) и «семантическая». А умственное действие по переводу новой лексической единицы из «эпизодической» памяти в «семантическую» в явном виде операционазировано и реализовано во «внешне-материальной» форме (известная терминология «теории планомерного формирования» П. Я. Гальперина) в виде диалоговой процедуры «тематической классификации».

Пользователь ЛЕКСИТРАНА опять же не принужден «исповедовать» предложенную ему автором категориальную иерархию: в любой момент он может внести новые мезо- и макрокатегории, убрать «авторские», то есть работать так, как ему удобно. И в этой его собственной творческой активности и реализуется та самая категориальная работа, которая и дает наиболее надежный мнемический эффект (эффект прочного запоминания лексики).

На основе категориальной иерархии TOPICS в ЛЕКСИТРАНЕ реализован особый режим тестирования (без сообщения правильных ответов), по результатам которого пользователь может увидеть тестовый профиль своей лексической компетентности — а именно, какие макро- и мезокатегории (области лексики) у него отстают по уровню обеспеченности реальным словарным запасом. ЛЕКСИТРАН гибко генерирует тесты (с помощью особых универсальных алгоритмов) для любых макро- и мезокатегорий, которые могут быть созданы самим пользователем для его собственного материала.

Литература

1. *Аткинсон Р.* Человеческая память и процесс обучения.

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

3. *Громов Г. Р.* Персональные вычисления — новый этап информационных технологий. Микропроцессорные средства и системы. 1984. № 1. С. 37—50.

4. *Клацки Р.* Память человека: структура и процессы.— М.: Мир, 1978. 320 с.

5. *Ляудис В. Я., Тихомиров О. К.* Психология и практика автоматизированного обучения. Вопросы психологии. 1988. № 2. С. 9—12.

6. *Растригин Л. А., Эренштейн М. Х.* Адаптивное обучение с моделью обучаемого. Рига: Зинатне, 1988. 160 с.

7. *Растригин Л. А.* Компьютерное обучение и самообучение. Информатика и образование. 1991. № 6. С. 42—46.

8. *Савельев А. Я.* (ред.) Обучающие машины, системы и комплексы. Справочник. Киев: Вища школа, 1986. 303 с.

9. *Талызина Н. Ф., Габай Т. В.* Пути и возможности автоматизации учебного процесса. М.: Изд-во Моск. ун-та, 1977.

10. *Фомичева Ю. В., Шмелев А. Г., Бурмистров И. В.* Психологические корреляты увлеченности компьютерными играми. Вестник Моск. ун-та. Психология. 1991. № 3. С. 27—39.

11. *Шмелев А. Г.* Введение в экспериментальную психосемантику. М.: Изд-во Моск. ун-та, 1983. 158 с.

12. *Шмелев А. Г., Похилько В. И.* ТЕЗАЛ — автоматизированный тезаурус личностных черт. Информатика и образование. 1988. № 6. С. 69—

М. СТЕПАНОВ

Единый методический подход к задачам на построение сложных компьютерных изображений

В курсе информатики, как правило, рассматриваются задачи на построение довольно простых компьютерных изображений, например, таких как эллипс с осями, параллельными сторонам экрана, и графики элементарных функций. Однако, достаточно естественное обобщение этих задач может вызвать значительные затруднения у самых сильных учащихся. В случае эллипса таким обобщением является задача на построение наклонного эллипса, а в случае графика — задачи на построение изображений пространственных кривых и поверхностей. Но даже решив одну из этих задач, учащийся не получает метода решения другой.

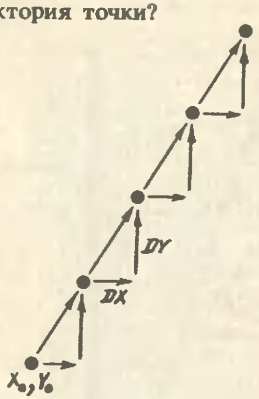
В данной статье предлагается единый методический подход к решению подобных задач, который разработан автором в рамках факультативного курса математического моделирования и является одним из фрагментов общей методики курса. Основная идея, обеспечивающая единство подхода, состоит в том, что геометрические формы трактуются как следы движущегося объекта. При этом все необходимые математические понятия вводятся неформально с опорой на моторную интуицию учащегося, что делает их вполне доступными. На основе приводимых ниже заданий можно вести самостоятельный факультативный курс. Для преподавателя полезно знакомство с работой [1].

Изложение ведется применительно к версии языка Бейсик, реализованной на ПЭВМ «Агат», но программы легко могут быть перенесены на другие компьютеры, предоставляющие доступ к графике. Задачи группируются в разделы, каждый из которых посвящен новым приемам построения изображений. В некоторых программах используются одни и те же подпрограммы, поэтому каждая такая подпрограмма получает имя и размещается в строках программы с фиксированными номерами.

1. Подготовительный этап. Параметрическое задание линий

Задание 1.1. Напишите программу, осуществляющую движение точки по экрану. При этом каждый раз точка должна смещаться вдоль осей X и Y на фиксированные расстояния DX и DY (рис. 1). Какова траектория точки?

1



Решение. Легко понять, что точка движется по прямой. Если DX и DY превосходят 1, то след точки является пунктиром.

```
10 HGR=1:COLOR=15:REM ВКЛЮЧЕНИЕ
                              ГРАФ.РЕЖ.
20 X0=30:Y0=40
30 DX=5:DY=4
40 N=40:REM КОЛИЧЕСТВО СКАЧКОВ
50 FOR T=0 TO N
60 PLOT X0+T*DX,Y0+T*DY:REM УСТАНОВКА
                                  ТОЧКИ
70 NEXT T
80 GET F$:TEXT=0:END:REM ЗАДЕРЖКА
                              ДО НАЖАТИЯ
```

Величина T из задания 1.1 может рассматриваться не только как номер очередной точки, но и как время, а величины DX и DY как горизонтальная и вертикальная составляющие скорости точки. Вычисление координаты этой точки производится по формулам:

$$\begin{aligned} X &= X0 + T \cdot DX \\ Y &= Y0 + T \cdot DY \end{aligned} \qquad (1)$$

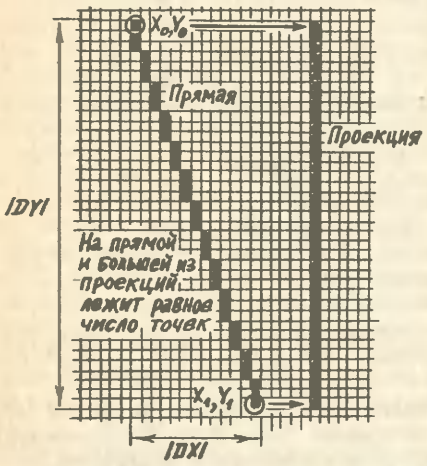
Задание 1.2. Напишите программу для проведения отрезков, концы которых могут лежать вне экрана.

Решение. Величина T в формулах (1) подобно физическому времени может принимать любые действительные значения, а не только натуральные. Предположим, что точ-

ка из положения (X0, Y0) должна за единицу времени перейти в положение (X1, Y1). Общее смещение точки по оси X при этом переходе равно DX=Y1—Y0. Легко проверить, что при T=0 и T=1 формула (1) дает нужные начальные и конечные координаты, а при промежуточных значениях T — координаты промежуточных точек отрезка.

Поскольку в программе переменная T не может быть непрерывной, необходимо

2



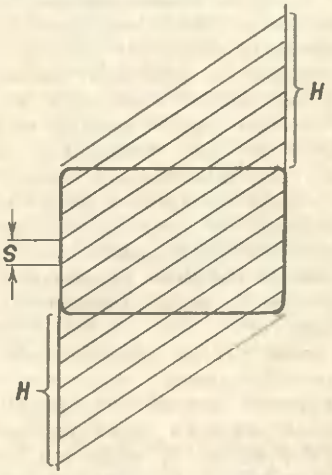
выбрать шаг, с которым будет меняться эта переменная. При этом следует исходить из того, что отрезок должен быть сплошным, а значит и его проекции на вертикаль и горизонталь тоже должны быть сплошными (рис. 2). Количество точек отрезка равно наибольшему из модулей чисел DX и DY. Прежде чем применить оператор установки точки, следует проверить, попала ли она в экран.

```
10 HGR=1:COLOR=15
20 X0=-10:Y0=-10
30 X1=265:Y1=265
40 GOSUB 2000
50 GET A$:TEXT=0:END
2000 REM ОТРЕЗОК
2005 REM ВХОД.ВЕЛ. X0,Y0,X1,Y1
2010 REM ПРОЧИЕ ВЕЛ. XN,YN,XK,YK,
                     DX,DY,NN,T,XT,YT
2015 XN=INT(X0):YN=INT(Y0)
2020 XK=INT(X1):YK=INT(Y1)
2025 DX=XK-XN:DY=YK-YN
2030 GOSUB 4000
2035 IF NN=0 THEN GOSUB 3000:RETURN
2040 FOR T=0 TO 1 STEP 1/NN
2045 XT=XN+T*DX:YT=YN+T*DY
2050 GOSUB 3000
2055 NEXT T
2060 RETURN
2065 REM КОНЕЦ
3000 REM ОБРАБОТКА КООРД.
3010 IF XT=>0 AND XT=<255 OR YT=>0
              OR YT=<255 THEN PLOT XT,YT
3020 RETURN
4000 REM ВЫБОР ШАГА
4010 NN=ABS(DX)
4020 IF NN<ABS(DY) THEN NN=ABS(DY)
4030 RETURN
```

Задание 1.3. Самостоятельно напишите программу для проведения отрезка-пунктира с заданным шагом.

Задание 1.4. Расчертите экран параллельными наклонными прямыми (рис. 3).

3



Решение.

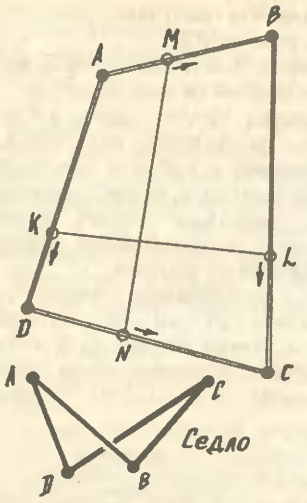
```
10 HGR=1:COLOR=15
20 X0=0:X1=255:H=145:S=20
30 FOR Y0=0 TO 255+H STEP S
40 Y1=Y0-H
50 GOSUB 2000
60 NEXT Y0
70 GET A$:TEXT=0:END
2000 REM ОТРЕЗОК
.........................
```

В задании 1.2 из формул (1) получены новые формулы, с помощью которых единичный отрезок отображается на произвольный отрезок с концами (X0, Y0) и (X1, Y1).

$$\begin{cases} X = X0 + T \cdot (X1 - X0) \\ Y = Y0 + T \cdot (Y1 - Y0) \end{cases} \qquad (2)$$

Задание 1.5. На экране задан четырехугольник ABCD. По его сторонам равномерно движутся 4 точки, каждая из которых проходит сторону за единицу времени. Точки M и N, K и L соединены резиновыми жгутами (рис. 4). Начертите на экране следы этих жгутов, оставляемые через равные промежутки времени.

Решение. Сначала построим следы, оставляемые жгутом MN, а затем аналогичным



способом — следы от жгута KL. Для одного и того же значения величины T по формулам (2) вычислим координаты точек M и N, а затем соединим эти точки отрезком.

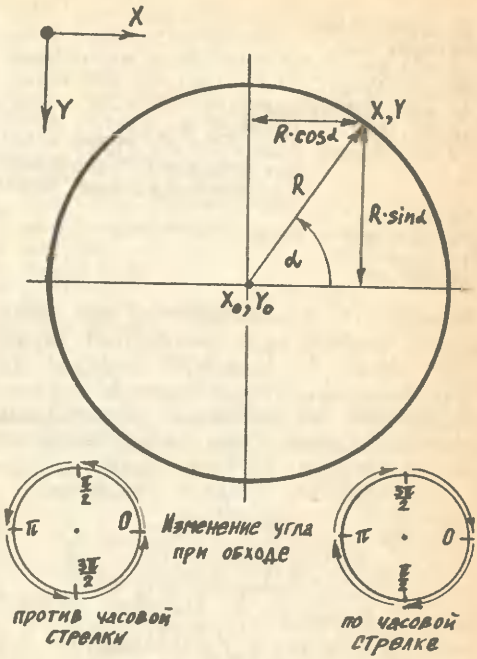
```
10 HGR=1:COLOR=15
20 DATA 10,10,160,170,240,0,80,190
30 N=16: REM КОЛ-ВО СЛЕДОВ ЖГУТА
40 READ XA,YA,XB,YB,XC,YC,XD,YD
50 FOR T=0 TO 1 STEP 1/N
60 X1=XA+T*(XB-XA):Y1=YA+T*(YB-YA)
70 X2=XD+T*(XC-XD):Y2=YD+T*(YC-YD)
80 PLOT X1,Y1 TO X2,Y2 :
                  REM ПРОВЕДЕНИЕ ОТРЕЗКА
90 NEXT T
100 FOR T=0 TO 1 STEP 1/N
110 X1=XA+T*(XD-XA):Y1=YA+T*(YD-YA)
120 X2=XB+T*(XC-XB):Y2=YB+T*(YC-YB)
130 PLOT X1,Y1 TO X2,Y2
140 NEXT T
150 GET F$:TEXT=0:END
```

При различном расположении вершин четырехугольника на экране может быть получено изображение седла или перекошенной решетки.

Для построения отрезка мы использовали формулы (2), в которых координаты точки, движущейся по отрезку, зависят от одной величины, называемой параметром. В формулах (2) зависимость является линейной. Если же использовать другие функциональные зависимости координат движущейся точки от параметра, можно получить вместо отрезков различные виды кривых. Параметр при этом можно воспринимать не только как время, но как величину, имеющую геометрический смысл.

Задание 1.6. Получите формулы для параметрического представления окружности и постройте окружность на экране.

Решение. Окружность на плоскости задается координатами центра и радиусом,



поэтому эти величины являются исходными. На роль параметра удобнее всего взять центральный угол. На рис. 5 легко усмотреть формулы

$$\begin{cases} X = X0 + R \cdot \cos\alpha \\ Y = Y0 - R \cdot \sin\alpha \end{cases} \qquad (3)$$

В языке Бейсик аргумент тригонометрических функций задается в радианах, поэтому мы будем использовать только радианную меру. При выводе формул (3) предполагалось, что угол растет при обходе окружности против часовой стрелки. Чтобы изменить направление обхода, достаточно изменить знак перед членом $R \cdot \sin\alpha$.

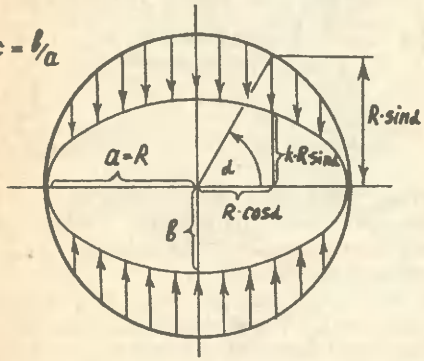
Поскольку экранные точки на «Агате» имеют форму прямоугольничков, а не квадратиков, то окружность на экране примет форму эллипса. Для корректировки изображения следует ввести коэффициент сжатия K. Подробнее об этом говорится в следующем задании.

```
10 HGR=1:COLOR=15:PI=4*ATN(1)
20 XO=128:YO=128:R=120:K=1
30 A0=0:A1=2*PI
40 GOSUB 2100
50 GET F$:TEXT=0:END
2100 REM ЭЛЛИПС
2105 REM ВХОД.ВЕЛ. XO,YO,R,A0,A1,K
2110 REM ПРОЧИЕ ВЕЛ. UG,DU,XT,YT
2115 GOSUB 4100
2120 FOR UG=A0 TO A1+.1 STEP DU
2125 XT=XO+R*COS(UG)
2130 YT=YO-K*R*SIN(UG)
```

```
2135 GOSUB 3100
2140 NEXT UG
2145 RETURN
2150 REM КОНЕЦ
3100 REM ОБРАБ.КООРД.
3110 IF UG=A0 THEN PLOT XT,YT
3120 IF UG<>A0 THEN PLOT TO XT,YT:
                REM ОТРЕЗОК ОТ ПОСЛЕДНЕЙ
                ПОСТАВЛЕННОЙ РАНЕЕ ТОЧКИ
3130 RETURN
4100 REM ВЫБОР ШАГА
4110 DU=PI/18
4120 RETURN
```

Задание 1.7. Исходя из того, что эллипс является сжатой (или растянутой) окружностью (рис. 6) выведите формулу для параметрического представления сжатого (растянутого) по вертикали эллипса. Подберите коэффициент сжатия так, чтобы эллипс на экране выглядел окружностью.

6



При сжатии вертикальные длины умножаются на коэффициент k

Решение. Окружность задается формулами (3). Из рис. 6 видно, что при вертикальном сжатии координата X не изменяется, а вертикальный отрезок длины $R \cdot \sin\alpha$ становится равным $K \cdot R \cdot \sin\alpha$. Отсюда получаем параметрическое задание эллипса

$$\begin{cases} X = X0 + R \cdot \cos\alpha = X0 + A \cdot \cos\alpha \\ Y = Y0 - K \cdot R \cdot \sin\alpha = Y0 - B \cdot \sin\alpha \end{cases} \qquad (4)$$

Формула соответствует программе «Эллипс» из задания 1.6. При $K>0$ построение эллипса производится против часовой стрелки, а при $K<0$ — по часовой. Для выбора значения коэффициента K, позволяющего получить окружность, используем программу

```
10 HGR=1:COLOR=15:PI=4*ATN(1)
20 XO=120:YO=120:R=50
30 A0=0:A1=2*PI
40 FOR K=1/3 TO 5/3 STEP 1/6
50 GOSUB 2100
60 NEXT K
70 GET F$:TEXT=0:END
2100 REM ЭЛЛИПС
.....................................
```

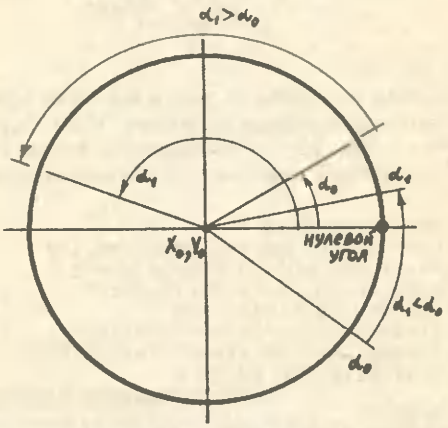
В результате получаем, что искомое значение K равно 5/3.

Задание 1.8. Разберитесь, как построить дугу окружности или эллипса.

Решение. Чтобы задать дугу, необходимо указать на эллипсе две точки, начальную и конечную, и выбрать направление обхода. Концы дуги мы будем задавать углами $\alpha 0$ и $\alpha 1$ (что уже учтено в подпрограмме «Эллипс»), а направление обхода — знаком коэффициента сжатия.

При организации цикла угол должен возрастать от $\alpha 0$ до $\alpha 1$, однако, в тех случаях, когда внутрь дуги попадает точка, задаваемая нулевым углом (рис. 7), может оказаться, что $\alpha 1 < \alpha 0$. Чтобы исправить

7



положение, нужно заменить угол $\alpha 1$ на угол $\alpha 1 + 2\pi$, при этом предполагается, что исходные значения углов $\alpha 0$ и $\alpha 1$ заключены между 0 и $2 \cdot \pi$. В дальнейшем мы будем использовать подпрограмму «Эллипс» и для углов, выходящих за эти границы. При этом дуга может быть несколько раз закручена вокруг эллипса, например, при $\alpha 0 = 0$ и $\alpha 1 = 10 \cdot \pi$ дуга делает 5 оборотов. В таких случаях мы будем следить за тем, чтобы $\alpha 0$ было меньше $\alpha 1$.

Выбор шага производится без учета радиуса, что неоправданно замедляет работу программы для маленьких окружностей. Попытайтесь усовершенствовать подпрограмму «Выбор шага».

```
10 HGR=1:COLOR=15:PI=4*ATN(1)
20 XO=120:YO=120:R=50
30 A0=3*PI/2:A1=PI/2
40 FOR K=0 TO -5/3 STEP -1/3
50 GOSUB 2100
60 NEXT K
70 GET F$:TEXT=0:END
2100 REM ЭЛЛИПС
.....................................
3100 REM ОБРАБ.КООРД.
.....................................
```

```
4100 REM ВЫБОР ШАГА
4110 IF A1<A0 THEN A1=A1+2*PI
4120 NN=INT((A1-A0)*18/PI)+1
4130 DU=(A1-A0)/NN
4140 RETURN
```

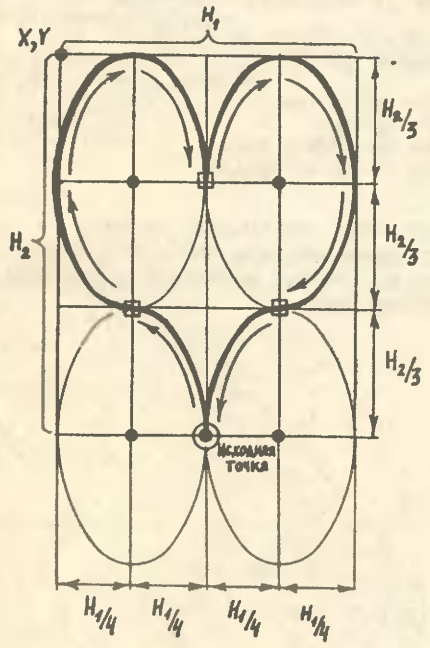
Нами получены параметрические представления отрезка, окружности и эллипса. Используя другие функциональные зависимости координат X и Y от параметра, можно получить представления различных кривых. В частности, график функции $y=f(x)$ может быть задан параметрически

$$x=t$$
$$y=f(t)$$

Существует и еще один способ параметрического задания кривых — составление кривой из дуг окружностей и прямолинейных отрезков. Кривая складывается из кусков, каждый из которых имеет свою параметризацию. Подобное задание кривой назовем кусочной параметризацией.

Задание 1.9. С помощью кусочной параметризации постройте сердечко.

8

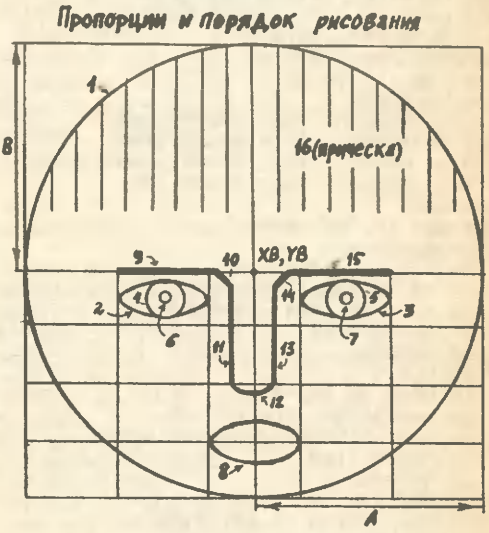


Решение. На рис. 8 представлено разбиение сердечка на 4 дуги равных между собой эллипсов. У всех эллипсов полуоси равны H1/4 и H2/3. Точно так же, в долях H1 и H2, будем выражать координаты их центров. Углы будем выражать в долях л. Направление обхода дуги зададим с помощью величины SK, равной либо +1, либо —1.

```
10 HGR=1:COLOR=15:PI=4*ATN(1)
20 X=5:Y=5:H1=245:H2=245
30 R=H1/4:KK=(H2/3)/(H1/4)
40 DATA .25,1,0,.5,1
50 DATA .25,.33,.5,2,-1
60 DATA .75,.33,1,.5,-1
70 DATA .75,1,.5,1,1
80 FOR I=1 TO 4
90 READ DX,DY,A0,A1,SK
100 XO=X+DX*H1:YO=Y+DY*H2
110 A0=PI*A0:A1=PI*A1
120 K=KK*SK
130 GOSUB 2100
140 NEXT I
150 GET F$:TEXT:END
2100 REM ЭЛЛИПС
.....................
```

Задание 1.10. Постройте сложное изображение, состоящее из дуг и отрезков (рис. 9).

9



Решение. Основной проблемой является унификация обращения к подпрограммам. Мы решаем ее следующим образом. Для каждого элемента изображения буквой русского алфавита указывается его тип (Э — эллипс, П — прямолинейный отрезок, Д — дуга). В зависимости от типа специальными подпрограммами производится считывание информации и ее преобразование, после чего используется подпрограмма построения соответствующего элемента.

Буквой «С» помечаются специализированные подпрограммы, которые связаны с особенностями именно этого изображения. В нашем случае используется специализированная подпрограмма построения прически.

Может показаться, что использование подпрограммы «Отрезок» только замедляет работу программы, ведь прямые можно

сразу провести оператором PLOT. Однако в дальнейшем мы воспользуемся тем, что в подпрограммах «Отрезок» и «Эллипс» можно перехватывать промежуточные координаты и преобразовывать их.

```
10 HGR=1:COLOR=15:PI=4*ATN(1):KE=16
20 XB=128:YB=96:A=64:B=80
30 DATA "З",0,0,1,1.25
40 DATA "З",-.4,.09,.2,.08
50 DATA "З",.4,.09,.2,.08
60 DATA "З",-.4,.09,.08,.08
70 DATA "З",.4,.09,.08,.08
80 DATA "З",-.4,.09,.04,.04
90 DATA "З",.4,.09,.04,.04
100 DATA "З",0,.67,.2,.08
110 DATA "П",-.6,0,-.14,0
120 DATA "П",-.14,0,-.07,.058
130 DATA "П",-.07,.058,-.07,.5
140 DATA "Д",0,.5,.07,.02,1,2
150 DATA "П",.07,.5,.07,.058
160 DATA "П",.07,.058,.14,.0
170 DATA "П",.14,0,.6,0
180 DATA "С"
190 FOR I=1 TO KE
200 READ PR$
210 IF PR$="П" THEN GOSUB 1000
220 IF PR$="З" THEN GOSUB 1100
230 IF PR$="Д" THEN GOSUB 1200
240 IF PR$="С" THEN GOSUB 1300
250 NEXT I
260 GET F$:TEXT=0:END
1000 REM П
1010 READ X0,Y0,X1,Y1
1020 X0=XB+X0*A:Y0=YB+Y0*B
1030 X1=XB+X1*A:Y1=YB+Y1*B
1040 GOSUB 2000
1050 RETURN
1100 REM З
1110 READ XO,YO,AA,BB
1120 XO=XB+XO*A:YO=YB+YO*B
1130 R=AA*A:K=BB/AA:A0=0:A1=2*PI
1140 GOSUB 2100
1150 RETURN
1200 REM Д
1210 READ XO,YO,AA,BB,A0,A1
1220 XO=XB+XO*A:YO=YB+YO*B
1230 R=AA*A:K=BB/AA
1240 A0=A0*PI:A1=A1*PI
1250 GOSUB 2100
1260 RETURN
1300 REM С
1310 Y1=YB-B*SIN(PI/18)
1320 FOR AL=PI/9 TO 8*PI/9 STEP PI/18
1330 X0=XB+A*COS(AL):Y0=YB-B*SIN(AL)
1340 X1=X0
1350 GOSUB 2000
1360 NEXT AL
1370 RETURN
2000 REM ОТРЕЗОК
.....................................
2100 REM ЭЛЛИПС
.....................................
```

2. Эллипс и построение объемных изображений

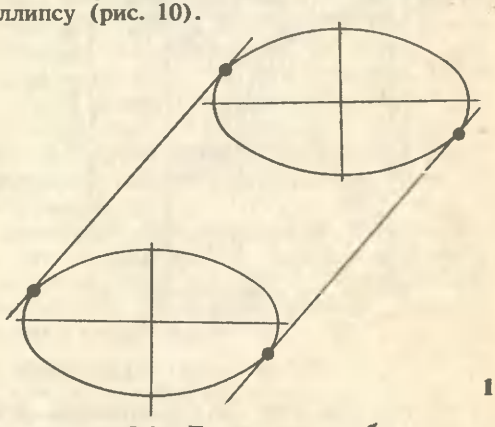
Если вращать твердое тело вокруг некоторой оси с тем, чтобы выбрать подходящий ракурс, любая его точка будет описывать круговую траекторию. Иногда по одной окружности могут двигаться сразу несколько важных для формы тела точек (например, вершин многогранника). Часто удобно строить изображение тела на плоскости, отправляясь именно от этих круговых траекторий. Поскольку окружность, как правило, кажется наблюдателю эллипсом, эллипс и его параметрическое представление приобретают важную роль при построении объемных изображений.

Задание 2.1. Постройте изображение прямого кругового цилиндра. Можно ли аналогичным способом построить изображения наклонного цилиндра и конуса?

Решение. Изменим подпрограмму обработки координат эллипса.

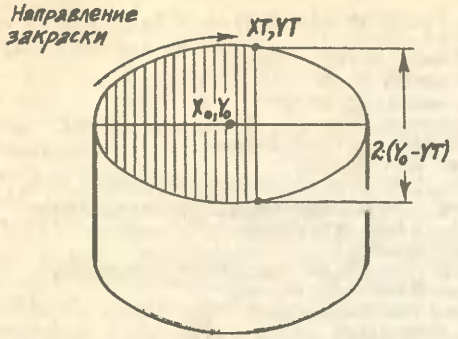
```
10 HGR=1:COLOR=15:PI=4*ATN(1)
20 XO=128:YO=50:R=100:K=-.5:H=90
30 A0=0:A1=2*PI
40 GOSUB 2100
50 GET F$:TEXT=0:END
2100 REM ЭЛЛИПС
.....................................
3100 REM ОБРАБОТКА КООРДИНАТ ЭЛЛИПСА
3110 IF UG=A0 THEN XP=XT:YP=YT
3120 IF UG=A0 THEN PLOT XT,YT
                    TO XT,YT+H:RETURN
3130 IF ABS(UG-PI)<.01 THEN
                    PLOT XT,YT TO XT,YT+H
3140 PLOT XT,YT TO XP,YP
3150 IF UG<PI THEN
                    PLOT XT,YT+H TO XP,YP+H
3160 XP=XT:YP=YT
3170 RETURN
4100 REM ВЫБОР ШАГА
4110 DU=PI/18:RETURN
```

Построить наклонный цилиндр и конус без дефектов мы пока не можем, так как не умеем проводить касательных к эллипсу (рис. 10).



10

Задание 2.1. Для того чтобы придать рисунку бóльшую выразительность, элементы изображения закрашивают. Закрасьте верхнее основание цилиндра, не прибегая к помощи специального оператора закраски, даже если он реализован на вашем компьютере.



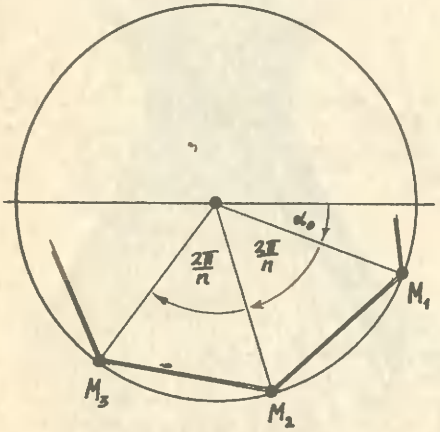
11

Решение. Будем закрашивать эллипс вертикальными прямыми (рис. 11). Для этого введем в программу из задания 2.1 подпрограмму «Отрезок» с измененной подпрограммой обработки координат отрезка. Изменим и дополним подпрограмму обработки координат эллипса со строки 3140.

```
3000 REM ОБРАБОТКА КООРДИНАТ ОТРЕЗКА
3010 PLOT XT,YT TO XT,YT+2*(YO-YT)
3020 RETURN
.....................................
3140 IF UG<PI THEN PLOT XT,YT TO XP,YP
3150 IF UG<PI THEN
                  PLOT XT,YT+H TO XP,YP+H
3160 IF UG<=PI THEN XP=XT:YP=YT: RETURN
3165 X0=XP:Y0=YP
3170 X1=XT:Y1=YT
3175 GOSUB 2000
3180 XP=X1:YP=Y1
3185 RETURN
```

Задание 2.3. Постройте изображение правильного n-угольника, вписанного в окружность.

12



Решение. Построение правильного n-угольника ничем кроме шага, с которым меняется угол, не отличается от построения окружности или эллипса. В случае эллипса сжат будет и многоугольник.

```
10 HGR=1:COLOR=15:PI=4*ATN(1)
20 XO=128:YO=128:R=72:K=-5/3
30 A0=PI/16:A1=A0+2*PI:N=5
40 GOSUB 2100
50 GET F$:TEXT=0:END
2100 REM ЭЛЛИПС
.....................................
3100 REM ОБРАБОТКА КООРДИНАТ ЭЛЛИПСА
3110 IF UG=A0 THEN PLOT XT,YT
3120 IF UG<>A0 THEN PLOT TO XT,YT
3130 RETURN
4100 REM ВЫБОР ШАГА
4110 DU=2*PI/N:RETURN
```

Задание 2.4. Постройте изображение усеченной пирамиды.

Решение. На экране будем строить все ребра пирамиды, в том числе и невидимые. Вершины пирамиды лежат на двух эллипсах, поэтому основания могут быть построены по образцу задания 2.3.

Поскольку при соединении вершин многогранников могут возникнуть определенные сложности, начиная с этого задания мы будем сначала вычислять координаты всех вершин, сохраняя их в массивах, а затем соединять их ребрами.


```
10 HOME:PRINT "СЧЕТ";
                          REM ОЧИСТКА ЭКРАНА
20 N=5:DIM XH(N),YH(N),XL(N),YL(N)
30 NM=0:REM ИНДЕКС МАССИВА
40 DATA 150,50,50
50 DATA 105,150,100
60 PI=4*ATN(1):A0=PI/6:A1=A0+2*PI:K=-.5
70 FOR I=1 TO 2
80 READ XO,YO,R
90 GOSUB 2100
100 NEXT I
110 HGR=1:COLOR=15
120 GOSUB 5000
130 GET F$:TEXT=0:END
2100 REM ЭЛЛИПС
.....................................
3100 REM ОБРАБОТКА КООРДИНАТ ЭЛЛИПСА
3110 IF I=1 THEN XH(NM)=XT:YH(NM)=YT
3120 IF I=2 THEN XL(NM)=XT:YL(NM)=YT
3130 NM=NM+1:IF NM>N THEN NM=0
3140 RETURN
4100 REM ВЫБОР ШАГА
4110 DU=2*PI/N:RETURN
5000 REM ПИРАМИДА
5010 FOR NM=0 TO N-1
5020 PLOT XH(NM),YH(NM)
                    TO XH(NM+1),YH(NM+1)
5030 PLOT XL(NM),YL(NM)
                    TO XL(NM+1),YL(NM+1)
5040 PLOT XH(NM),YH(NM)
                        TO XL(NM),YL(NM)
5050 NEXT NM
5060 RETURN
```

Задание 2.5. Удалите из изображения пирамиды невидимые ребра.

Решение. Мы не можем сколько-нибудь подробно обсуждать вопросы удаления невидимых элементов изображения. Продемонстрируем только самый простой способ,

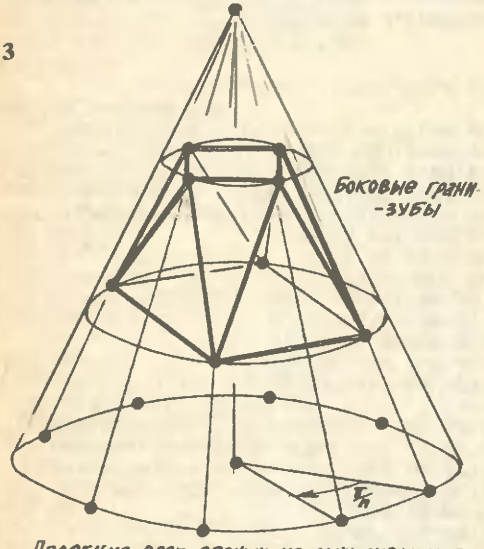
применимый к изображениям многогранников. Назовем его «ручным» удалением элементов. С учетом положения исходной вершины и направления обхода эллипса установим индексы массивов, соответствующие ребрам, а затем наложим условия на построение ребер.

```
5030 IF NM<>2 AND NM<> 3 AND NM<>4
              THEN PLOT XL(NM),YL(NM)
                   TO XL(NM+1),YL(NM+1)
5040 IF NM<>3 AND NM<>4
              THEN PLOT XH(NM),YH(NM)
                        TO XL(NM),YL(NM)
```

Аналогичным способом можно заменить сплошные изображения невидимых элементов на пунктирные, использовав при этом модифицированную подпрограмму «Отрезок».

Задание 2.6. Постройте изображение многогранника (рис. 13), который мы будем называть n-угольным зубоскалом (он является частным случаем призматоида).

13



Решение. Зубоскал имеет $2 \cdot n$ вершин. Их положение в совокупности определяется углами, кратными π/n. В отличие от усеченной пирамиды, координаты вершин зубоскала не будем разбивать на две группы. Это позволит легко провести все ребра: боковые — соединят соседние в массиве вершины, а лежащие в основании — через одну.

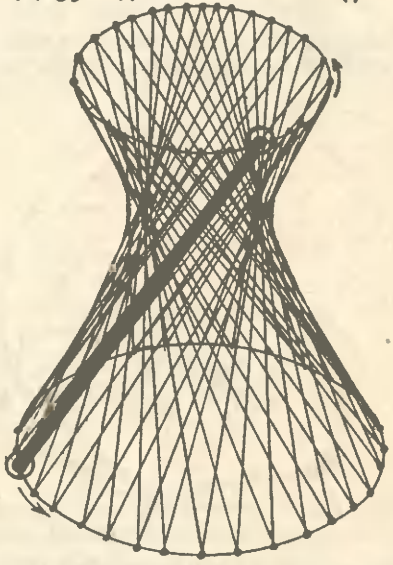
```
10 N=5
20 DIM X(2*N),Y(2*N):NM=0
30 R=50:R1=70:H=100:K=-.5:XO=128:YO=30
40 PI=4*ATN(1):A0=PI/6:B=1
50 A1=A0+2*PI
60 HOME:PRINT "СЧЕТ"
70 GOSUB 2100
80 HGR=1:COLOR=15
90 GOSUB 5000
100 GET F$:TEXT=0:END
2100 REM ЭЛЛИПС
.....................................
3100 REM ОБРАБОТКА КООРДИНАТ ЭЛЛИПСА
3110 X(NM)=XT:Y(NM)=YT
3120 NM=NM+1
3130 REM СМЕНА ОСНОВАНИЯ ЗУБОСКАЛА
3140 RR=R:R=R1:R1=RR
3150 YO=YO+B*H:B=-B
3160 RETURN
4100 REM ВЫБОР ШАГА
4110 DU=PI/N:RETURN
5000 REM ЗУБОСКАЛ
5010 FOR NM=0 TO 2*N-1
5020 PLOT X(NM),Y(NM)
                    TO X(NM+1),Y(NM+1)
5030 PN=NM+2:IF PN>2*N THEN PN=1
5040 PLOT X(NM),Y(NM) TO X(PN),Y(PN))
5050 NEXT NM
5060 RETURN
```

Задание 2.7. Изменив предыдущую программу, постройте зубчатую звездочку. Указание. Достаточно положить H=0, Y0=128, K=—5/3 и убрать строки 5030 и 5040.

Задание 2.8. Концы отрезка скользят по двум окружностям, лежащим в параллельных плоскостях. Постройте изображение поверхности, заметаемой отрезком (рис. 14).

„Закрученный" зубоскал — однополостный гиперболический параболоид

14



Решение. Используем программу из задания 2.7, удалив строки 5030 и 5040, а также заменив строки

```
10 N=32:M=19
50 A1=A0+2*PI/M
4110 DU=PI*M/N:RETURN
```

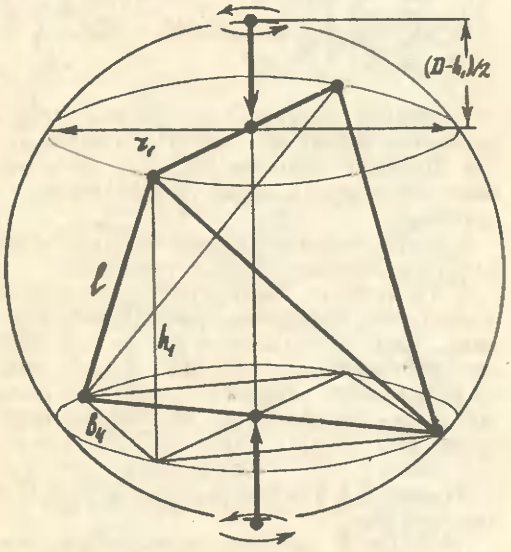
3. Построение изображений, связанных с шаром

Пять правильных многогранников (других не существует), изображенных на рисунках 15—19, называются Платоновыми телами, по имени философа Платона, который считал, что атомы четырех элементов, признаваемых древними греками, имеют форму тетраэдра (огонь), октаэдра (воздух), икосаэдра (вода) и куба (земля), а Вселенная в целом имеет форму додекаэдра. С тех пор Платоновы тела прочно вошли в культуру человечества. Их рисовали Леонардо и Дюрер. Эта же задача стоит и перед нами.

3.1. Выбор оси вращения

Основным инструментом при построении Платоновых тел остаются круговые траектории, заметаемые вершинами многогранника при вращении вокруг некоторой оси. По понятным причинам, оси, не проходящие через центр описанного вокруг многогранника шара, следует сразу же отбросить. Таким образом для определения положения оси достаточно указать помимо центра шара еще одну принадлежащую ей точку. После этого вершины разделятся на группы, каждая из которых принадлежит одной круговой траектории, образуемой сечением шара плоскостью, перпендикулярной оси и изображаемой на рисунке в виде эллипса. Рассмотрим три варианта выбора оси (рис. 15—19).

15 Тетраэдр



Порядок вычислений

$z_1 = l/2$

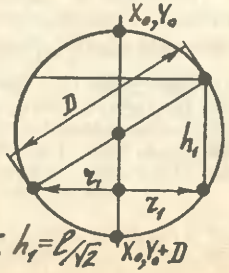
$b_4 = 2 \cdot z_1 \cdot \sin\frac{\pi}{3} = l/\sqrt{2}$

$h_1 = \sqrt{l^2 - b_4^2} = l/\sqrt{2}$

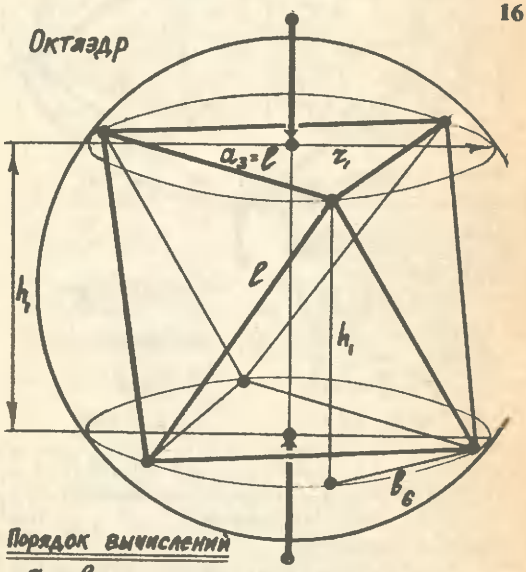
$D = \sqrt{4z_1^2 + h_1^2} = \sqrt{\frac{3}{2}}\,l$

Результаты

$l = \sqrt{\frac{2}{3}}\,D$; $z_1 = \frac{l}{2}$; $h_1 = l/\sqrt{2}$



16



Порядок вычислений

$a_3 = l$

$z_1 = a_3/2 \cdot \sin\frac{\pi}{3} = l/\sqrt{3}$

$b_6 = 2 \cdot z_1 \cdot \sin\frac{\pi}{6} = l/\sqrt{3}$

$h_1 = \sqrt{l^2 - b_6^2} = \sqrt{\frac{2}{3}}\,l$

$D = \sqrt{4z_1^2 + h_1^2} = \sqrt{2} \cdot l$

Результаты

$l = D/\sqrt{2}$

$z_1 = l/\sqrt{3}$

$h_1 = \sqrt{\frac{2}{3}}\,l$

1. Ось проходит через середину ребра (тетраэдр).

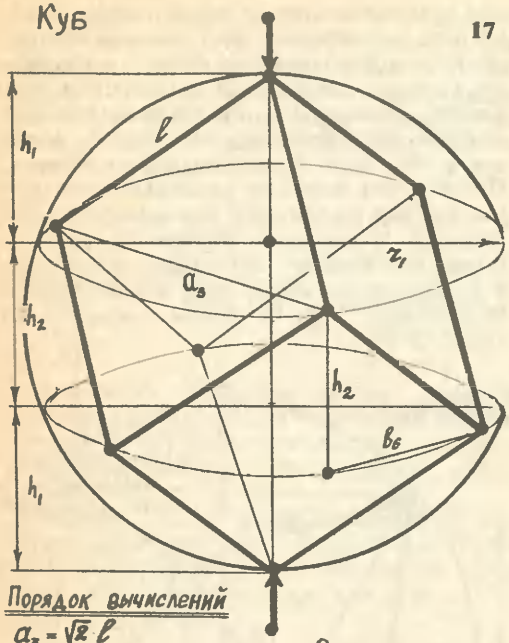
2. Ось проходит через вершину (куб, икосаэдр).

3. Ось проходит через центр грани (октаэдр, додекаэдр).

Обычно для куба выбирают третий вариант, а для тетраэдра и октаэдра — первый. Мы же остановились на более экзотическом их расположении.

Плоскости, в которых лежат траектории вершин, рассекают Платоновы тела на уже знакомые нам многогранники, открывая путь к построению нужных изображений.

1. Тетраэдр = двуугольный зубоскал (см. задание 2.6).



Порядок вычислений

$a_3 = \sqrt{2}\,\ell$

$r_1 = a_3/2 \cdot \sin\frac{\pi}{3} = \sqrt{\frac{3}{4}}\,\ell$

$h_1 = \sqrt{\ell^2 - r_1^2} = \ell/\sqrt{3}$

$b_6 = 2 \cdot r_1 \cdot \sin\frac{\pi}{6} = r_1$

$h_2 = \sqrt{\ell^2 - b_6^2} = \ell/\sqrt{3}$

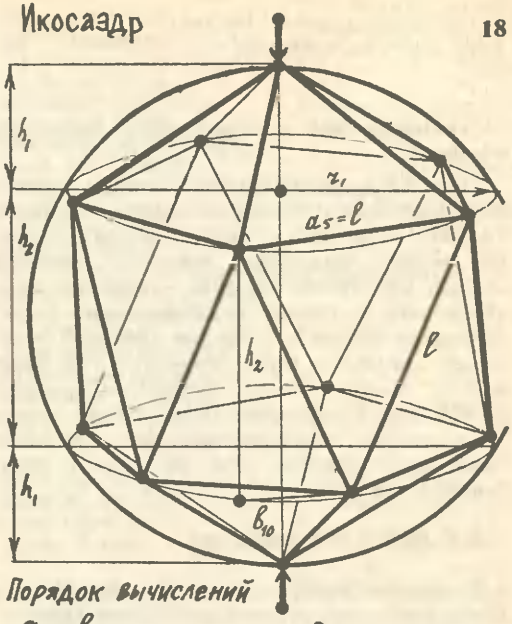
$D = 2h_1 + h_2 = \sqrt{3} \cdot \ell$

Результаты

$\ell = D/\sqrt{3}$

$r_1 = \sqrt{\frac{3}{4}} \cdot \ell$

$h_1 = h_2 = D/3$



Порядок вычислений

$a_5 = \ell$

$r_1 = \ell/2 \cdot \sin\frac{\pi}{5} \approx 0{,}85\ell$

$b_{10} = 2 \cdot r_1 \cdot \sin\frac{\pi}{10} \approx 0{,}525\ell$

$h_1 = \sqrt{\ell^2 - r_1^2} \approx 0{,}525\ell$

$h_2 = \sqrt{\ell^2 - b_{10}^2} \approx 0{,}85\ell$

$D = \sqrt{4r_1^2 + h_1^2} \approx 1{,}9\ell$

Результаты

$\ell = 0{,}525D$

$h_1 = 0{,}525\ell$

$h_2 = 0{,}85\ell$

$r_1 = 0{,}85\ell$

2. Октаэдр = треугольный зубоскал.

3. Куб = треугольный зубоскал + 2 треугольных пирамиды.

4. Икосаэдр = пятиугольный зубоскал + + 2 пятиугольных пирамиды.

5. Додекаэдр = пятиугольный зубоскал + + 2 усеченных пятиугольных пирамиды.

3.2. Метрические соотношения для Платоновых тел

Высоты пирамид и зубоскалов, радиусы эллипсов не могут быть выбраны произвольно, так как внешний вид многогранника исказится. Нужно выразить все расстояния, характеризующие многогранник, через одну исходную величину. В качестве таковой мы возьмем диаметр описанного шара.

В процессе вычислений мы будем использовать известную формулу, выражающую зависимость между радиусом окружности и стороной вписанного в нее многоугольника:

$$a_n = 2 \cdot r \cdot \sin \pi/n.$$

Все вычисления и их результаты приведены на рис. 15—19.

Распишем порядок построения многогранников, общий для всех Платоновых тел.

1. Выбрать величины X0, Y0, D, задающие положение и длину вертикальной оси вращения.

2. Вычислить координаты центров эллипсов и их большие полуоси (радиусы).

3. Произвести вычисление и сохранение в массивах координат вершин многогранника. При этом следует учесть тот факт, что все вершины, лежащие выше центра шара, сдвинуты относительно вершин, лежащих ниже центра шара, подобно вершинам зубоскала.

Задание 3.3. Постройте изображения Платоновых тел.

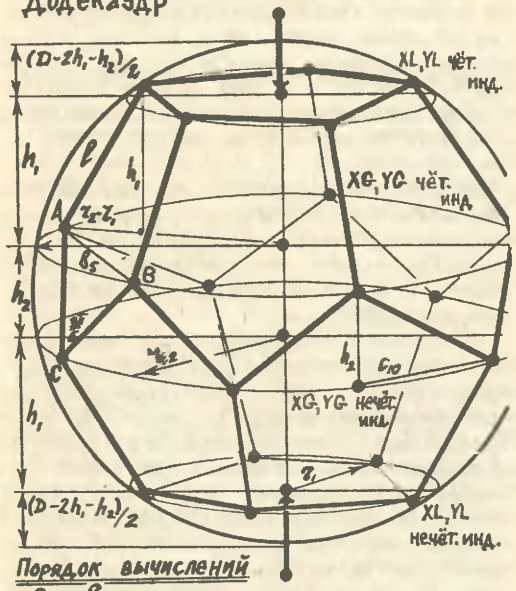
Решение. В качестве примера продемонстрируем построение додекаэдра (без использования подпрограмм).

```
10 HGR=1:COLOR=15:PI=4*ATN(1)
20 X0=128:Y0=5:K=.25:I=0
30 D=180:A0=PI/4:DU=PI/5
40 L=D*.356:H1=L*.85:H2=L*.525
50 R1=L*.85*.6:R2=L*1.38*.6:
        REM КОЭФ. .0.6 УЧИТЫВАЕТ
                        СЖАТИЕ ЭКРАНА
60 Y1=Y0+(D-2*H1-H2)/2
70 Y2=Y1+H1:Y3=Y2+H2:Y4=Y3+H1
```

```
80 DIM XL(10),YL(10),XG(10),YG(10)
90 FOR UG=A0 TO A0+2*PI+.1 STEP DU
100 XL(I)=X0+R1*COS(UG)
110 YL(I)=Y1-K*R1*SIN(UG)
120 YY=Y1:Y1=Y4:Y4=YY
130 XG(I)=X0+R2*COS(UG)
140 YG(I)=Y2-K*R2*SIN(UG)
150 YY=Y2:Y2=Y3:Y3=YY
160 I=I+1
170 NEXT UG
180 FOR I=0 TO 9
190 J=I+2:IF J>10 THEN J=J-10
200 IF I<>1 AND I<>9 THEN
        PLOT XL(I),YL(I) TO XL(J),YL(J)
210 IF I<>0 AND I<>1 AND I<>2 THEN
        PLOT XL(I),YL(I) TO XG(I),YG(I)
220 IF I=>3 AND I<=8 THEN
    PLOT XG(I),YG(I) TO XG(I+1),YG(I+1)
230 NEXT I
240 GET F$:TEXT=0:END
```

19

Додекаэдр



Порядок вычислений

$a_5 = \ell$

$r_1 = \ell/2 \cdot \sin\frac{\pi}{5} \approx 0{,}85\ell$

$b_5 = 2\ell \cdot \sin\frac{2\pi}{10} \approx 1{,}62\ell$ (ΔABC)

$r_2 = b_5/2 \cdot \sin\frac{\pi}{5} \approx 1{,}38\ell$

$h_1 = \sqrt{\ell^2 - (r_2 - r_1)^2} \approx 0{,}85\ell$

$c_{10} = 2 \cdot r_2 \cdot \sin\frac{\pi}{10} \approx 0{,}85\ell$

$h_2 = \sqrt{\ell^2 - c_{10}^2} \approx 0{,}525\ell$

$D = \sqrt{4r_2^2 + h_2^2} \approx 2{,}8\ell$

Результаты

$\ell = 0{,}356\,D$

$r_1 = 0{,}85\ell$

$r_2 = 1{,}38\ell$

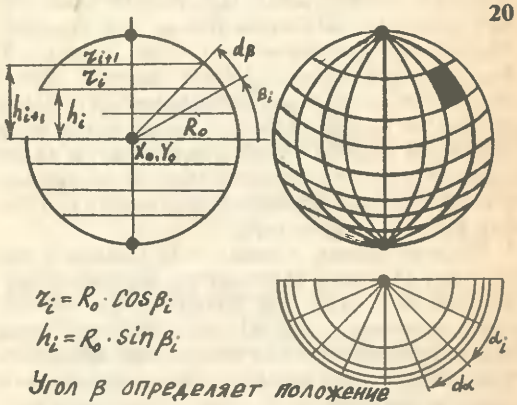
$h_1 = 0{,}85\ell$

$h_2 = 0{,}525\ell$

Задание 3.4. Постройте изображение сферы, подчеркнув ее форму меридианами и параллелями.

Решение. Заменим сферу вписанным в нее многогранником с большим количеством граней. Для этого разрежем ее на слои горизонтальными плоскостями (рис. 20). Каждый слой заменим правильной усеченной $2 \cdot n$-угольной пирамидой. Боковые грани этих пирамид покроют поверхность сферы. Положение слоя определяется углом β, а положение грани в слое — углом α.

20



$r_i = R_0 \cdot \cos\beta_i$

$h_i = R_0 \cdot \sin\beta_i$

Угол β определяет положение и размеры эллипсов, а угол α положение вершин на эллипсе

45

```
10 HGR=1:COLOR=15:PI=4*ATN(1)
20 X0=128:Y0=128:R0=120:K1=.6:K=-.5
30 N=18:DIM R(N),H(N),
                X1(N),Y1(N),X2(N),Y2(N)
40 FOR I=0 TO N
50 R(I)=R0*K1*COS(-PI/2+I*PI/N)
60 H(I)=R0*SIN(-PI/2+I*PI/N)
70 NEXT I
80 FOR A=0 TO PI+.1 STEP PI/N
90 C=COS(A):S=SIN(A)
100 FOR I=0 TO N
110 X2(I)=X0+R(I)*C
120 Y2(I)=Y0+H(I)-R(I)*K*S
130 IF I=0 THEN PLOT X2(I),Y2(I)
140 IF I<>0 THEN PLOT TO X2(I),Y2(I)
150 NEXT I
160 FOR I=1 TO N-1
170 IF A=0 THEN 190
180 PLOT X1(I),Y1(I) TO X2(I),Y2(I)
190 X1(I)=X2(I):Y1(I)=Y2(I)
200 NEXT I
210 NEXT A
220 GET F$:TEXT=0:END
```

Литература

1. *Бескин Н. М.* Изображение пространственных фигур.— М.: Наука, 1971.

Е. ЧУБРОВ, С. ИВАНОВ

ФОРМУЛА — инструмент математического эксперимента

Компьютер по-настоящему придет в школу только тогда, когда он будет использоваться в других дисциплинах. Программистами станут единицы, а компьютером, как профессиональным инструментом в различных областях, пользоваться будут многие. Главное — это, во-первых, понимание того, какие задачи на данном рабочем месте могут быть поручены электронному помощнику, а какие должны остаться за человеком и, во-вторых, умение критически проанализировать результат работы компьютера.

На этих идеях основан наш подход к определению места компьютера на уроках математики. С ним мы познакомили читателей журнала в № 3—4, 1992 г. Здесь будет подробно рассказано о том направлении, которое мы называем «математическим экспериментом».

Инструментом исследования математических объектов для наших учащихся служит программа ФОРМУЛА. Мы начнем с описания возможностей этой программы, а в следующей публикации остановимся на методике проведения занятий.

ФОРМУЛА используется на компьютерных уроках по математике в старших классах, а также на младших курсах технического вуза. При правильном выборе тем эффект от применения компьютера может быть значительным. Впрочем, познакомившись с ФОРМУЛОЙ, и школьники и студенты очень быстро понимают ее преимущества и начинают самостоятельно использовать ее для разных целей. Программа позволяет легко произвести вычисления (даже весьма громоздкие), построить на экране и распечатать графики. Ее достоинства — очень простое управление, наглядность, богатый набор вычислительных возможностей.

ФОРМУЛА работает на любом IBM-совместимом компьютере с монитором типа EGA или VGA. Ниже мы даем обзор функций программы, чтобы можно было представить, как она применяется для организации компьютерных уроков.

ФОРМУЛА имеет два режима — Режим Вычислений и Режим Построения Графиков.

Запись формул. Чтобы вычислить какое-либо выражение, нужно ввести его с клавиатуры. Вводить математические формулы можно в строчку, используя пять знаков операций (+ — * /^), круглые скобки, и обозначения встроенных функций.

ФОРМУЛА вычислит его и результат появится на экране.

ФОРМУЛА допускает более «математическую» форму записи формул, в которой можно использовать три уровня записи: основной, верхний (для показателей степени) и нижний (для основания логарифмов и индексов). Кроме того, вы можете обозначать извлечение квадратного корня символом $\sqrt{}$, а абсолютную величину в виде $|...|$.

(1, 2, 3), (sin (x), cos (x), (1,0), (0,1) — тоже формулы. Они называются *списками*. Списки по своей сути похожи на массивы, но они могут содержать все что угодно — числа, другие списки, в общем, любые формулы. Списки позволяют организовать вычисления с векторами, матрицами, комплексными числами и т. п.

Переменные и команды. При вычислениях обычно нужно запоминать промежуточные результаты. Кроме того, когда много раз считаешь по одной и той же *формуле*, удобно ввести эту формулу один раз, а потом использовать многократно.

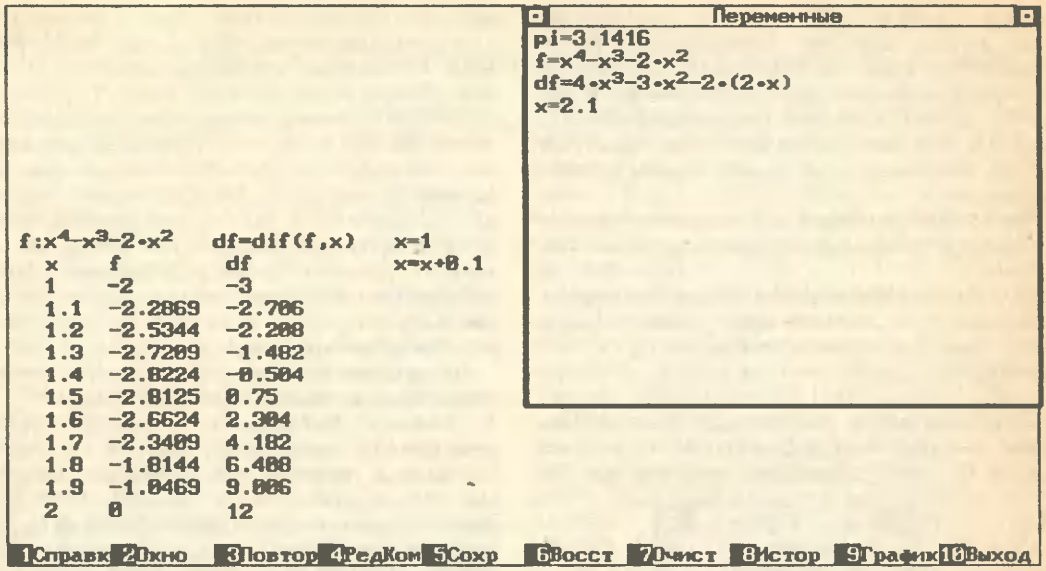
В ФОРМУЛЕ значением переменной может быть произвольная математическая формула. При этом число рассматривается как простейший вид формулы. Когда в команде присваивания стоит двоеточие, то выражение из правой части без всяких изменений присваивается указанной переменной. Такая команда называется «ПРИСВОИТЬ». Когда в команде стоит знак равенства, то выражение в правой части сначала вычисляется, и только затем результат вычисления присваивается переменной. Такая команда называется «ВЫЧИСЛИТЬ и ПРИСВОИТЬ». Наконец, вы можете просто ввести какую-нибудь формулу и нажать Enter. Введенная формула вычисляется и результат выводится на экран (но не присваивается никакой переменной). Такая команда называется «ВЫЧИСЛИТЬ».

Все переменные и значения, которые им присвоены, показаны в верхней части экрана (*окне переменных*). А ход диалога, то есть ваши команды и ответы ФОРМУЛЫ на них, отображаются в нижней части экрана (*поле диалога*).

Пример. Построим таблицу значений функции $x^4-x^3-2\cdot x$ и ее производной на отрезке [1, 2] с шагом 0.1. Задачу решают следующие команды.

```
f:x^4-x^3-2·x    df=dif (f, x)    x=1
f    df          x=x+0.1
```

1



Повторить последнюю команду 10 раз.

Результат показан на рис. 1. Из него хорошо видно, что там, где функция проходит через минимум (между 1.4 и 1,5), производная меняет знак с минуса на плюс.

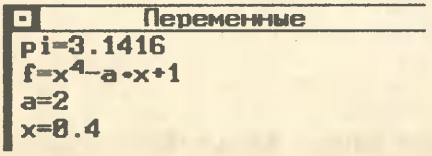
Как вы видели из примера, можно вводить по несколько команд на одной строке, разделяя их пробелами, а внутри каждой команды пробелов быть не должно.

Механизм вычислений — самое важное в работе ФОРМУЛЫ. Он достаточно прост, но несколько необычен. Ведь значением переменной может быть целая формула, чего не бывает ни в БЕЙСИКЕ, ни в ПАСКАЛЕ. На самом деле, подобный подход обычно применяется в системах, манипулирующих с формулами, таких как REDUCE или MACSYMA. Необходимость такого же подхода в учебных программах по математике представляется нам очевидной.

Когда же ФОРМУЛА включает свой механизм вычислений? При выполнении команды «ВЫЧИСЛИТЬ» или «ВЫЧИСЛИТЬ И ПРИСВОИТЬ», а также при анализе выражения, задающего график, и при вводе границ диапазона координат графического окна.

Как ФОРМУЛА вычисляет? Представим себе, что у вас определены переменные, показанные на рис. 2. Введем команду «ВЫЧИСЛИТЬ» вида $\sqrt{f}$. ФОРМУЛА должна вычислить это выражение и результат вывести на экран. Так как использована переменная (f), то сначала проверяется, определена ли она. f определена. Поэтому для вычисления $\sqrt{f}$ нужно вычислить f и подставить результат. Но для вычисления f нужно использовать переменные a и x, которые тоже определены и имеют числовые значения. Поэтому все вычисление даст числовой результат 0.47497.

2



А если бы некоторые переменные отсутствовали, например, не было бы x? В обычных языках программирования это означает ошибку программиста. В ФОРМУЛЕ не так. Если переменной нет, то ее значение просто не подставляется. В нашем случае на экране будет выведен ответ: $\sqrt{(x^2-2\cdot x+1)}$.

То, что мы описали, называется *принципом частичного вычисления*. Это означает, что выражение вычисляется настолько, насколько это возможно при данных значениях переменных.

Обратите внимание, что ФОРМУЛА «не заметила» полного квадрата и не упростила выражение до $|x-1|$. ФОРМУЛА не относится к *системам аналитических вычислений*, которые решают математические задачи за человека. Она разрабатывалась для других целей — как инструмент, используемый при изучении математики.

Есть еще две команды, которые называются «УДАЛИТЬ» и «ПОКАЗАТЬ». Команда «УДАЛИТЬ переменную x» имеет вид:

x=

Переменные нужно удалять, когда они больше не нужны, чтобы не загромождать окно переменных. Есть еще две ситуации,

когда команда «УДАЛИТЬ» необходима.

Во-первых, нельзя строить график функции от x, если переменная x определена. Если вы про это забыли, то можете получить совсем не то, что ожидали. ФОРМУЛА сначала вычислит вашу функцию, подставив в нее значение x, а потом будет строить график.

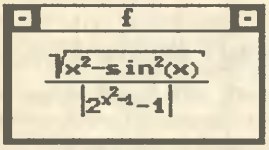
Аналогично, нельзя найти производную от формулы по переменной, которая имеет значение.

Команда «ПОКАЗАТЬ» поможет вам разобраться со сложными формулами. Например, если вы ввели такую команду «ПРИСВОИТЬ»:

f:$\sqrt{(x^2-\sin(x)^2)}/|2^{x\wedge 2-1}-1|$

и сомневаетесь в ее правильности, то введите команду «ПОКАЗАТЬ f» и увидите формулу в более привычном виде (см. рис. 3).

3



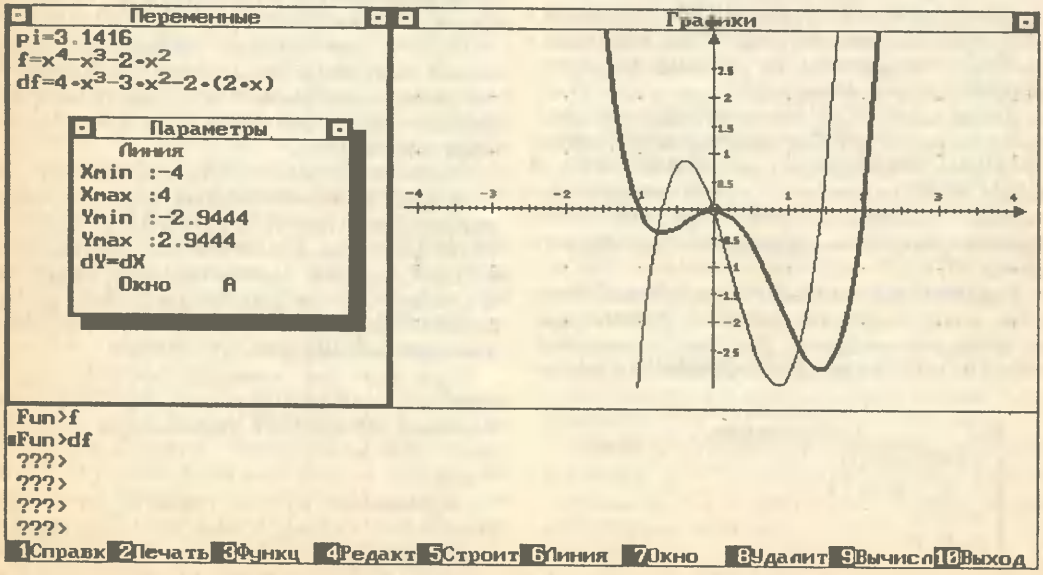
Другие возможности Режима Вычислений. Функциональные клавиши используются для вызова справочного режима, изменения размеров полей на экране, повторения последней команды, вызова ее для редактирования и последующего ввода. Можно записать на диск или восстановить с него список переменных с их значениями, удалить сразу все переменные (кроме pi), вызвать ИСТОРИЮ, т. е. список последних десяти команд, выполненных пользователем. Вы можете выбирать команды из этого списка для их повторного выполнения. Выбранную из ИСТОРИИ команду можно перед выполнением отредактировать.

ФОРМУЛА имеет встроенные функции для вычислений из самых разных разделов математики. Это — производная и определенный интеграл, частичная сумма ряда и датчик случайных чисел, разложение числа на множители и наибольший общий делитель. И, конечно, самые разнообразные элементарные и специальные функции — от квадратного корня и синуса до интеграла ошибок и полиномов Лежандра.

Построение графиков. Теперь обратимся к графическим возможностям ФОРМУЛЫ. В режиме Построения Графиков экран компьютера представлен на рис. 4. Здесь построены графики той функции, которую мы табулировали, и ее производной. Конечно, чтобы понять связь функции и ее производной, лучше один раз увидеть (графики), чем сто раз протабулировать (функции). Правда, численный метод может дать гораздо более точный результат...

Слева на экране — поле переменных и справа — окно графиков. Шесть строк в нижней части экрана — это поле определения графиков. На экране можно иметь до 6 кривых одновременно. В нашем примере мы построили два графика — функций d и df. Сами функции были определены ранее в Режиме Вычислений, и здесь мы просто вводим их обозначения. Здесь работает тот же самый механизм вычислений. Но можно определить графики и прямо в Режиме Построения Графиков.

4



Небольшое окно, наложенное на поле переменных,— это *окно параметров*, где показаны текущие диапазоны координат графиков и текущий цвет линии.

Типы графиков. Кроме графиков функций, заданных в форме f (x), ФОРМУЛА умеет строить графики еще пяти типов:

функцию, заданную в полярных координатах в виде $\rho(\psi)$;

кривую, заданную в параметрической форме $\{x(t), y(t), t_0 \leqslant t \leqslant t_1\}$;

линию уровня функции двух переменных $F(x, y) = C_1, C_2, \ldots$;

поле направлений, заданное функцией двух переменных F (x, y);

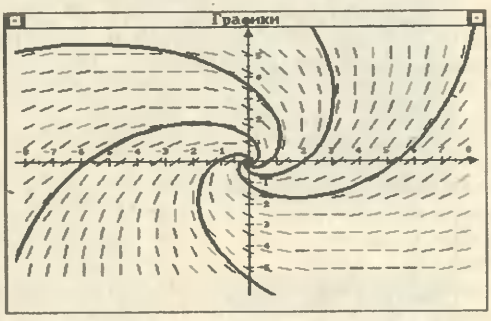
и просто ломаную линию, заданную своими вершинами:

$(x_1, y_1) \pm (x_2, y_2) \pm \ldots$ Плюс означает, что отрезок ломаной нужно провести, а минус — что нет.

Сначала вы выбираете в появившемся меню один из перечисленных выше типов и вводите аналитическое определение графика. Затем строите сам график. Можно отредактировать формулу, определяющую график, после чего тот автоматически перестраивается, удалить график и его определение, очистить окно полностью.

На рис. 5 показано поле направлений дифференциального уравнения $y' = (x-y)/(x+y)$ и несколько его интегральных кривых (заданных в полярных координатах).

5



Управление окном графиков. Для выбора диапазона координат той части координатной плоскости, которая отображена в окне, служит *окно параметров*. Вы переходите в это окно, изменяете границы окна (X_{min}, X_{max}, Y_{min}, Y_{max}) и ФОРМУЛА перестраивает графики в новом диапазоне координат.

Удобно также пользоваться опцией «dX = = dY». Если выбрать этот пункт меню, будут установлены разные масштабы по осям координат. После этого кривая с уравнением окружности на чертеже действительно будет круглой (при неравных масштабах она имеет вид эллипса).

Окно графиков можно двигать по координатной плоскости в четырех направлениях; уменьшать (увеличивать) вдвое диапазон по каждой из координат, не переходя в окно параметров.

Маркер. Его мы еще называем «черепахой», потому что это дальний родственник знаменитой черепахи Пейперта из языка Лого. Маркер появляется на экране в виде кружка с перекрестием. Вы можете двигать маркер по экрану клавишами со стрелками, и он будет показывать на своем индикаторе (в последней строке экрана) декартовы и полярные координаты своего центра. С помощью этого «устройства» легко определить координаты характерных точек, например, точки пересечения двух кривых. Кроме того, маркер может показать значение функции F (x, y) в данных точках плоскости (если вы работаете с линиями уровня или полем направлений). Скоростью движения маркера можно управлять клавишами «+» и «—».

В Режиме Построения Графиков можно вывести чертеж на принтер, развернуть окно графиков на весь экран, сменить графическое окно. Прежнее окно записывается на диске, к нему можно позже вернуться. Эта возможность позволяет педагогу подготовить заранее все необходимые чертежи, а ученикам загрузить их в нужный момент урока.

Предусмотрен выбор типа и цвета линии для очередного графика. В вашем распоряжении 13 цветов и три типа — толстая линия, тонкая и пунктир.

Есть вывод на экран меню, из которого можно выбрать тип координатных осей и вид их разметки. Разметка каждой из осей может быть десятичной или в долях π.

Ждем ваших откликов и предложений.

ПРОГРАММНЫЕ ПРОДУКТЫ ДЛЯ ВАС!

Advansed Instructional Software Trading AB представляет в странах бывшего СССР интересы фирмы IST AB - International Software Trading AB и предлагает программные продукты для системы народного образования.

TEACHCAD v3.5 - конструктор гиперсред. Не зная ни одного языка программирования, Вы сможете построить графическую базу данных, создать "живой" отчет, смоделировать какое-либо действие или лекцию по интересующей Вас теме. Создание нужной Вам программы перестанет быть проблемой и займет всего 10% Вашего времени. TeachCAD - это новая технология для Вас!

JUNIOR-1 v2.0 - набор развивающих игр для детей 7 - 12 лет, разбитый на три части разной сложности. Junior-1 это не просто программы! Это программно-методический комплекс, который можно использовать как в школе, так и дома. Перед ребенком ставятся задачи, в которых надо найти давно известные в мире взрослых решения, основанные на таких серьезных понятиях, как алгоритм, рекурсия, дихотомия, стек и т.д. Junior-1 поможет Вашим детям изобрести эти понятия самостоятельно!

FIVE STARS v1.0 - изящный интегрированный пакет, простой и мощный одновременно - специально для Вас и Ваших учеников! Вы научите детей пользоваться электронными таблицами и составите бюджет школы. С Вашей помощью они освоят базы данных и помогут решить Вам все административные проблемы. Вместе с детьми Вы погрузитесь в удивительный мир компьютерной графики и научитесь профессионально работать с клавиатурой. Современный редактор текстов поможет Вам прекрасно оформить любые документы. Five Stars - это маленький мир с большими возможностями!

	Коммерческая цена (USD)	Цена для системы образования (USD)
JUNIOR-1 v2.0	117	57
Часть 1	42	20
Часть 2	48	23
Часть 3	35	17
FIVE STARS v1.0	125	61
Клавиатурный тренажер	19	9
Графический редактор	25	12
Электронные таблицы	32	15
База данных	29	14
Текстовый редактор	32	15
TEACHCAD v3.5	99	49

Если Вас заинтересовали эти программные средства, обращайтесь, пожалуйста, к нашему дилеру - НЦПСО (Научный центр программных средств обучения).
Адрес: 109004, г.Москва, ул.Б.Коммунистическая, 9А, НЦПСО при МДО.
Телефон: (095) 272 26 71, Сергей Сергеевич Майоров.
р/с 609909 в Гагаринском коммерческом банке г.Москвы МФО 201315.

Все цены указаны в долларах США, но Вы можете заплатить рублями по текущему курсу ММВБ.

К. ШУЛЬМАН
координатор проекта «Образование» фирмы IBM

Проект «Пилотные школы»: год спустя

Чуть больше года прошло с момента завершения первого этапа проекта «Культура, Наука, Образование: СССР — США», в рамках которого развивался проект «Пилотные школы». Не такой уж большой срок для того, чтобы делать глобальные выводы, но вполне достаточный для подведения первых итогов и анализа результатов.

Пытаясь осмыслить роль этого проекта в развитии системы образования, мы в первую очередь должны понять те задачи, которые стояли перед участниками проекта. Понятно, что этот проект не мог решать задачи массовой компьютеризации народного образования. Примером тому — принцип распределения поступающей техники по школам. Чаще всего компьютерные классы устанавливались в тех школах, которые уже имели опыт использования вычислительной техники в учебном процессе. Приобретался этот опыт на технике отечественного производства — БК-0010, УКНЦ, «Корвет» и других типах компьютеров, имеющихся в школах к 1990 г.— времени начала реализации проекта.

Главная цель проекта «Пилотные школы» — разработка новых технологий применения вычислительной техники в образовании, и эту цель мы считаем успешно достигнутой.

При этом несколько сотен тысяч школьников (в проекте участвует чуть больше 1 % школ бывшего СССР) стали работать на современных, надежных и эргономических компьютерах.

Перед организаторами проекта стояло несколько достаточно сложных проблем, начинавшихся с обеспечения доставки вычислительной техники с завода фирмы IBM в Гриноке (Шотландия) и заканчивающихся организацией ее эффективного использования. Очевидно, что последняя задача является наиболее сложной. Во многом успех ее реализации связан с применением опыта, накопленного фирмой IBM в области образования и обучения.

Надо отметить, что IBM известна в мире не только как фирма, производящая вычислительную технику, а главным образом как поставщик комплексных решений, в которых вычислительная техника служит лишь одним из компонентов. Обязательным элементом комплексного решения является набор системного и прикладного программного обеспечения, организации обучения и технического обслуживания.

Подготовка специалистов для реализации проекта проводилась по схеме, традиционной для реализации больших проектов. Сначала фирмой IBM была подготовлена группа учителей, методистов и специалистов по обслуживанию вычислительной техники и установлены компьютерные классы в двух школах Москвы. Затем методисты Центра поддержки проекта начали обучение сотрудников региональных Центров поддержки проекта и одновременно с этим — разработку программно-методических комплексов (ПМК) по различным предметам. При этом программы, учебники и методические пособия, входившие в состав разрабатываемых ПМК, испытывались в двух школах. Благодаря такому тестированию в реальных условиях были получены программные и методические материалы, удачно вписывающиеся в реальный учебный процесс.

Таким образом, к моменту установки техники в школах региональных Центров

поддержки проекта уже были подготовлены учителя и программно-методические материалы по ряду предметов.

Вопрос с программно-методическим обеспечением участников проекта был (и остается), пожалуй, одним из самых острых, поскольку именно наличие программ и методик определяет эффективность всего проекта. Сотни курсов по разным предметам можно найти в каталогах программ, издаваемых образовательными подразделениями фирмы IBM в США, Великобритании и других странах. Однако их применение весьма проблематично из-за ориентации на иные системы образования, с одной стороны, и трудностями с локализацией — с другой.

Говорят, что есть два способа накормить человека: можно дать ему ведро рыбы, а можно дать удочку и научить ловить рыбу. Такой «удочкой» стал инструментальный пакет LinkWay — специализированное средство для создания компьютерных курсов. Этот пакет позволяет создавать интерактивные программы, сочетающие тексты, статические и движущиеся объекты, цвет и графику, звук и видеоизображения. Последняя версия этого пакета — IBM LinkWay Live! 1.0 — обладает расширенными сетевыми возможностями, позволяет использовать технологию DVI и поддерживает графический стандарт XGA.

Говоря о школах, участвующих в проекте «Пилотные школы», мы не должны забывать и о других участниках образовательного проекта: школа для детей с нарушениями слуха и зрения, детских садах, центрах продолженного инженерного образования, школах бизнеса и учреждениях профессиональной подготовки. Комплексные решения для этих участников проекта были определены с учетом их специфических потребностей. Так, для обучения детей с нарушениями слуха применяется программно-аппаратный комплекс «Видимая речь», позволяющий визуализировать звук с помощью компьютера и облегчить обучение речи глухих и слабослышащих детей. Центрами продолженного инженерного образования применяются системы «мультимедиа», благодаря которым реализуются возможности компьютера в сочетании с разработанными Международной Ассоциацией продолженного инженерного образования (IACEE) видеокурсами. В профессиональной подготовке используются технологические лаборатории и методы обучения Всемирного Союза ORT. Эти методы обучения базируются на принципе «изобрети изобретенное» и ориентированы на обучение специальностям, связанным с электроникой, волоконной оптикой, связью, пневматикой и робототехникой.

Немного статистики. На реализацию 1-го этапа проекта уже ушло почти два года (март 1990 г.— декабрь 1991 г.). За это время организовано и оснащено свыше 40 региональных Центров поддержки, оборудованы компьютерные классы в 1000 общеобразовательных школах и 37 школах для детей-инвалидов, открыто 15 школ бизнеса и 13 Центров продолженного инженерного образования, оснащено несколько организаций профессиональной подготовки и детских садов. Разработаны десятки программно-методических комплексов, несколько тысяч учителей прошло базовую и специализированную подготовку, установлено почти 14 000 персональных систем IBM PS/2 различных моделей.

Многие из читателей ИНФО, как участвующие в проекте «Пилотные школы», так и узнающие о нем сегодня, наверняка пытались понять, почему именно фирма IBM стала партнером Гособразования СССР по реализации этого проекта? Ведь наверняка те 19 миллионов долларов, затраченные на реализацию 1-го этапа проекта «Культура, Наука, Образование: СССР — США», можно было потратить на приобретение более дешевых компьютеров других производителей и тем самым оснастить большее количество школ. Или вложить эти деньги в развитие отечественной промышленности, установив в школах КУВТы на основе БК-0010 и УКНЦ. Вот лишь несколько примеров, подтверждающих обоснованность этого выбора.

1. «Price versus Cost». Будем сравнивать не начальные цены компьютеров разных производителей, а стоимость обладания компьютером в течение его жизни (5—6 лет). Более надежный и, следовательно, более дорогой компьютер одного производителя может оказаться более выгодной покупкой, чем более дешевый, но менее надежный — другого. Более низкая цена обычно рассматривается как плата за риск — риск не иметь возможности для гарантийного ремонта или риск не найти запасные части для ремонта через 2—3 года. На складах фирмы IBM запасные части к компьютерам хранятся в течение 7 лет, что является своеобразной страховкой на случай поломки.

2. «Сравнивайте яблоки и яблоки». К тому моменту, когда IBM полностью перешла на выпуск мониторов, поддерживающих режим VGA, большинство поставщиков предлагали мониторы, поддерживающие режим EGA. Такой монитор стоит дешевле, чем VGA, при этом он значительно более вреден для глаз.

3. «Эргономика и гигиеничность». Приобретая компьютер, мало кто задумывается

над эргономичностью клавиатуры, основным фактором которой чаще всего является наличие русских букв. Независимые исследования показывают, что клавиатуры фирмы IBM оказываются на 7—10 % более производительными, чем клавиатуры других фирм (вспомним пленочную клавиатуру БК-0010 и больные пальцы учеников!). Такие гигиенические факторы, как уровень шума и радиации обычно не учитываются при выборе компьютера, а зачастую эта информация просто отсутствует в описании. Однако эти факторы могут оказаться критическими при длительной эксплуатации. Заметим лишь, что мониторы фирмы IBM соответствуют самым жестким в мире стандартам качества: более того, зачастую эти стандарты устанавливаются на основе эксплуатационных характеристик компьютеров фирмы IBM.

Вот далеко не полный перечень аргументов в пользу IBM. Чтобы более полно представить роль IBM в этом проекте добавим к этому активное (в том числе финансовое) участие фирмы в развитии программы экономического образования школьников «Junior Achivement» и соглашение с фирмой BORLAND о предоставлении скидки всем участникам проекта при покупке программного обеспечения фирмы.

Каково же состояние проекта «Пилотные школы» на сегодняшний день? Можно ли считать его завершенным? Десятки классов на базе нового типа персональных систем — IBM PS/1 поступили в СНГ в 1992 году. Финансируемые как за счет спонсоров (заводов, банков), так и за счет средств, заработанных школами и вузами самостоятельно (!), эти классы расширяют круг участников проекта «Пилотные школы». Многие региональные центры стали деловыми партнерами фирмы IBM, финансируя образовательные программы за счет средств, получаемых от продажи комплексных технологий IBM. В середине ноября в Киевском Политехническом институте состоялось открытие Центра компьютерных интегрированных технологий. Этот центр оснащен самыми современными компьютерами, использующими RISC-процессоры, и создан для переподготовки инженеров в области компьютерного автоматизированного проектирования и производства.

Экономическая ситуация, сложившаяся в странах СНГ, определяет особенности развития проекта в ближайшем будущем. Основное внимание уделяется контактам с учебными заведениями, что привело к распространению на территории СНГ специальной образовательной инициативы IBM. В этой программе предусмотрена поставка по льготным ценам вычислительной техники для использования в процессе обучения и управления образованием что, как мы надеемся, позволит большему числу школ, ПТУ и других учебных заведений стать участниками образовательных проектов фирмы IBM и получить все преимущества от такого сотрудничества.

Разумеется, не все проходило гладко и не везде удалось избежать ошибок. Так, например, возможности инструментального пакета Private Tutor, входившего в комплект школьного класса на первом этапе проекта, оказались недостаточными для большинства участников проекта, и в комплект поставки был введен LinkWay. Поступало много предложений оснастить рабочие места учеников жесткими дисками, и теперь все ученические рабочие места имеют винчестеры 40 Мбайт.

Разумеется, это — лишь часть пожеланий участников проекта, реализованных фирмой IBM в течение 1992 г. Мы знаем специфические потребности учебных заведений и надеемся, что наши комплексные решения сумеют помочь достигнуть цели, стоящей перед системой образования.

Мы благодарны журналу «Информатика и образование» и его читателям за активное участие в реализации и развитии проекта «Пилотные школы» и будем рады ответить на письма и звонки подписчиков журнала.

Предложения, замечания и вопросы просим направлять в представительство фирмы IBM в Москве:

113054, Москва, ул. Бахрушина, 18.
Телефоны: 235-66-02, 235-48-44,
телефакс 235-49-49.

РЕГИОНАЛЬНЫЙ ЦЕНТР «ЛОГОСЪ»

Предлагает пользователям УК-НЦ, IBM PC, «Агат», БК-0010, «Корвет» комплексные информационные услуги в 1993 г.

Почти наверняка Вы уже пользовались нашим программным обеспечением, но только зарегистрированные пользователи получают фирменное сопровождение.

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. Мы с удовольствием вышлем Вам подробный каталог программ бесплатно.

Мы сотрудничаем с ведущими заводами-изготовителями ВТ в СНГ и поможем Вам улучшить техническое состояние Вашего класса, отремонтировав и модифицировав технику, установив сетевое оборудование.

Вы можете 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.

10 магнитных дискет в течение года. Стоимость подписки на 20 % ниже, чем суммарная стоимость входящих в нее программных продуктов.

Подписка может быть оформлена в течение всего 1993 года. Стоимость подписки — 15 000 руб.

Условно бесплатное информационное обслуживание

Для частных лиц и организаций мы продолжаем 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 на 6-ти магнитных дискетах, оплатив только стоимость дискет и накладные расходы.

Подписка может быть оформлена в течение всего 1993 года. Стоимость подписки — 1500 руб.

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.

Вы также можете оформить заказ, позвонив нам по телефону.

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.

Звоните нам по телефонам: (8350) 24-81-09, 22-82-18.

Пишите нам по адресу: 428034, г. Чебоксары, а/я 99.

Наши официальные реквизиты: 428017, г. Чебоксары, пр. М. Горького, д. 5. Региональный центр «ЛОГОСЪ». Р/с 200467497 в Коммерческом банке «Чувашсоцбанк», МФО 286017.

Е. ПОЛАТ

Телекоммуникации в школе

Азы телекоммуникации

Какую роль могут играть в педагогической практике телекоммуникации?

Если бы педагогика как наука могла и умела делать запросы другим областям знания, в первую очередь технике и технологии, то, возможно, дела в школе были бы лучше. Что имеется в виду?

Ну, например, учителя давно заметили, что классная доска при всей своей универсальности и непререкаемом авторитете для школьных будней не вполне удобна для преподавателя.

Во-первых, учитель вынужден подчас довольно долгое время находиться к классу спиной, если он пишет на доске. Во-вторых, не каждый учитель обладает четким почерком и навыками рисования или черчения. В-третьих, процедура письма мелом на доске занимает массу времени. В-четвертых, сама эта деятельность мало эстетична, если учитель пользуется обычным мелом.

И вот сформировался определенный педагогический запрос — создать в помощь учителю техническое средство, обладающее дидактическими свойствами и функциями классной доски, но лишенное указанных недостатков.

Технический прогресс не заставил себя долго упрашивать. Есть решение — графопроектор. А дальше уже дело учителей, ученых-методистов разрабатывать педагогически целесообразную систему его применения, создать условия реализации его свойств, отвечающие тем или иным методическим задачам, т. е. дальше снова возникает педагогическая задача интеграции этого технического средства в практику школы и создания соответствующего средства обучения — транспарантов разных видов и назначений.

Другой пример с лингафоном. Запрос также исходил от педагогики. Надо было достаточно быстро и эффективно обучать людей говорить на иностранном языке и понимать иноязычную речь. Причем имелось в виду не индивидуальное, а групповое обучение. Были созданы языковые лаборатории разных типов активного пользования. А дальше опять задача педагогов — разработать достаточно эффективную методику использования этого технического средства, с одной стороны, и адекватные средства обучения, которые позволили бы реализовывать на практике все дидактические свойства лингафона, с другой. В разных странах к этой проблеме подошли по-разному. Оттого и различные результаты. Но разве виновато само техническое средство? Конечно, нет. Все дело в последующем педагогическом решении.

Эти рассуждения небеспочвенные, когда мы обращаемся в настоящее время к компьютерной технике и пытаемся интегрировать ее в педагогический процесс.

В практике, к сожалению, приведенные выше примеры, как хорошо знает читатель, единичны. Чаще всего педагоги идут как бы вслед за научно-техническим прогрессом, оценивая в педагогической практике возможности применения того или иного технического средства, разработанного специалистами для других целей. Так было с учебным кино, радио, телевидением. Так же обстоит дело и с компьютером. Может быть именно в этом и кроется весьма часто наблюдаемая неэффективность применения этих технических средств, в том числе и компьютера в целях обучения и воспитания. Ибо сначала было Слово... Сначала следует внятно сказать, что нам нужно, чтобы... Чего мы хотим? Какие решаем задачи? Если мы хотим воспитать послушного исполнителя, то вполне достаточно таких технических средств обучения, которые, с одной стороны, позволяли бы жестко регламентировать и дозировать информацию, предназначенную для ученика, а с другой, не имели бы эффективной обратной связи.

Если же мы хотим воспитать человека, самостоятельно творчески мыслящего, умеющего находить и рационально решать проблемы, нам потребуются иные средства обучения, иные методические и педагогические системы.

Компьютер дает возможность максимально индивидуализировать обучение, сделать процесс обучения творческим, исследовательским. Готовы ли мы к этому педагогически?

Видимо, здесь уместно будет заметить, что, учитывая практические потребности специалистов различных профессий, в школе важно приобрести некоторые базовые навыки работы с компьютером, а именно:

— различные способы введения информации через клавиатуру, с помощью различного рода манипуляторов типа «мышь»;

— пользование текстовым редактором;

— пользование электронными таблицами;
— создание и пользование базами и банками данных;
— пользование телекоммуникацией;
— распечатка на принтере;
— некоторые операции с винчестером и гибкими дисками.

Но это лишь одна сторона дела, хотя и весьма существенная. Другая заключается в том, чтобы ясно осознавать, какое педагогическое содержание вкладывается в процесс освоения знания тем или иным свойством технического средства.

Важно четко представить, какие педагогические, дидактические задачи наиболее актуальны в наши дни с учетом дня завтрашнего? Какие из них по своим дидактическим особенностям адекватны дидактическим свойствам компьютера вообще и свойствам телекоммуникации, в частности.

Итак, попробуем выделить некоторые из педагогических и дидактических задач, имея в виду возможность использования для их решения телекоммуникации.

Необходимо отметить, что школьное общение происходит в достаточно замкнутом социуме, в котором очень многое зависит от общего культурного, нравственного уровня входящих в него людей. Кроме того, следует учитывать и то, что существующая система образования, полностью основанная на классно-урочной системе, авторитарна по самой своей сути. Слово учителя — закон для учащихся. Хорошо, если в школе дружный, грамотный коллектив, обладающий высокой культурой и профессиональной компетентностью. А если это не так? Как выйти ребятам, да и учителям из этого замкнутого социума? Телекоммуникация может по своим свойствам представить условия, возможности для хотя бы частичного решения этой проблемы.

Как, каким путем — это дело педагогов, методистов, которые должны разработать педагогическое содержание, максимально раскрывающее потенциальные дидактические свойства этого нового технического средства. Педагоги должны определить его дидактические функции.

В настоящее время авторитарная педагогика изжила себя. Учитель не должен быть ментором, подавляющим своей властью волю и интеллект учащихся. Учитель в современной школе и школе будущего — это старший товарищ и наставник, который организует и ненавязчиво направляет самостоятельную деятельность учащихся. Каждому ученику любого возраста необходимо дать шанс попробовать свои силы в разных областях деятельности. Человек в период своего становления должен иметь возможность попробовать свои силы в различных социальных ролях — и в роли лидера, генератора идей, и в роли координатора, и в роли исполнителя, способного грамотно выполнить поручение. Разнообразие ролевой деятельности учащихся в процессе обучения — аспект психологии в педагогике, еще мало изученный, но весьма перспективный. Телекоммуникации, если в основу их использования положен метод проектов, о чем подробнее будет рассказано в последующих публикациях, телеконференции также могут оказаться весьма полезными, незаменимыми в этом вопросе. Наконец, гуманитарное, языковое развитие ребят — тоже немаловажный фактор, способствующий общему культурному и интеллектуальному развитию человека.

Все эти педагогические задачи могла бы помочь решить компьютерная телекоммуникация — электронная почта, телеконференции. Каковы же ее дидактические функции или другими словами, где можно обозначить ее место в школьной практике?

Уже накопленный, хотя пока и не очень большой опыт применения телекоммуникаций в различных сферах образования показал, что этот вид информационных технологий позволяет:

— организовать различного рода совместные исследовательские работы учащихся, учителей, студентов, научных работников из различных школ, научных и учебных центров одного или разных регионов или даже разных стран. Метод проектов позволяет при этом организовать подлинно исследовательскую творческую либо чисто прикладную практическую самостоятельную деятельность партнеров, используя при этом многообразие методов и форм самостоятельной познавательной и практической, творческой работы;

— организовать оперативную консультационную помощь широкому кругу обучаемых из научно-методических центров;

— организовать сеть заочного обучения и повышения квалификации педагогических кадров;

— оперативно обмениваться информацией, идеями, планами по интересующим участников совместных проектов вопросам, темам, расширяя таким образом свой кругозор, повышая свой культурный уровень;

— формировать у партнеров, кто бы они ни были — учащиеся, студенты, учителя — коммуникационные навыки, культуру общения, что предполагает со стороны партнеров умения кратко и четко формулировать собственные мысли, терпимо относиться к мнению партнеров, умение вести дискуссию, аргументированно доказывать свою точку

зрения и уметь слушать и уважать мнение партнера;

— формировать навыки подлинно исследовательской деятельности, моделируя работу научной лаборатории, творческой мастерской и т. д.

На наш взгляд, телекоммуникации при грамотном их использовании позволяют осуществить принципиально новый подход к обучению и воспитанию учащихся. Этот подход:

— базируется на широком обучении, сближении, стирании границ между отдельными социумами; на свободном обмене мнениями, идеями, информацией участников совместного проекта, на вполне естественном желании познать новое, расширить свой кругозор;

— имеет в своей основе реальные исследовательские методы (научная и или творческая лаборатории), позволяющие познавать законы природы, основы техники, технологии, социальные явления в их динамике, особенности разнообразных видов творчества в процессе совместной деятельности группы участников;

— основан на широких контактах с культурой других народов, опытом других людей;

— естественным образом стимулирует развитие гуманитарного образования, акцентирует внимание на нравственных аспектах жизни и деятельности человека, на состоянии и сохранении окружающей его среды;

— стимулирует развитие как родной речи участников сотрудничества, так и овладение иностранными языками, если речь идет о международных проектах;

— способствует приобретению как учащимися, так и учителями разнообразных сопутствующих навыков, которые могут оказаться весьма полезными в последующей жизни, в том числе и навыков пользования компьютерной техникой.

Разумеется, успех телекоммуникационных проектов во многом зависит от ряда факторов:

— от надежности и возможностей используемой техники;

— от подлинного интереса всех участников совместного проекта к избранной теме, что, в свою очередь, в немалой степени зависит от практической ценности этого проекта;

— от удачного подбора участников, который должен быть сугубо добровольным, полностью ориентированным на личную заинтересованность участников проекта в данной теме;

— от удачного выбора лидеров проекта;

— наконец, от общей координации работы со стороны координатора и учителей, практического внедрения полученных результатов. Думается, что читателю понятно, что речь идет об органичной интеграции телекоммуникационных проектов в общую педагогическую концепцию, в учебно-воспитательный процесс, где наряду с этим подходом и в сочетании с ним используются разнообразные методы и средства обучения, в том числе и традиционные.

Таковы педагогические возможности, дидактические функции телекоммуникаций для целей образования.

М. КОВТУН

Программирование в системе Express Pascal

Массивы и строки. Операторы цикла. Оператор перехода

Один из самых распространенных видов работы, выполняемых компьютером — просмотр и обработки больших наборов однотипных данных.

Рассмотрим простейшую задачу такого сорта — ввести с клавиатуры 10 чисел и найти среди них максимальное.

Первое, что нам нужно сделать для этого — завести переменные, в которых мы сможем разместить введенные числа. (Есть, конечно, возможность решить эту задачу без накопления всех чисел в памяти — выбирать максимальное число по мере ввода. Но такое решение невозможно, если нужно не найти максимальное среди введенных чисел, а упорядочить введенные числа или как-нибудь по-другому их обработать. Поэтому мы рассмотрим решение, в котором все числа сначала размещаются в памяти, а потом среди них имеется максимальное.)

Ясно, что заводить 10 переменных плохо. Ну, с десятью мы еще справимся, а как быть, если потребуется 100 или 1000? Вместо этого нужно завести одну переменную, тип которой — массив чисел. Описание нужной нам переменной имеет вид:

```
var  xx : array [1..10] of real;
```

Здесь впервые после двоеточия появилось не имя типа, а описание типа. Описание состоит из зарезервированного слова array означающего, что переменная этого типа является массивом, диапазона изменения индексов в квадратных скобках (от 1 до 10), зарезервированного слова of и следующего за ним типа элементов массива.

Продолжение. Начало см.: Информатика и образование. 1992. № 1—6.

Можно считать, что данное описание определяет сразу десять переменных типа real с именами xx [1], xx [2], ..., xx [10]. Число, указываемое в квадратных скобках после имени переменной-массива, называется индексом. Вместо числа может быть указано произвольное выражение целого типа.

Первое, чего мы добились с помощью типа массива — это одним махом описали сразу 10 переменных (и, при необходимости, с теми же затратами труда могли бы описать и 100, и 1000 переменных). Но, кроме этого количественного достижения, мы получили качественное увеличение возможностей — возможность динамического вычисления имени переменной в процессе работы программы. Ведь если где-то в программе мы напишем «xx [i]», то, в зависимости от текущего значения i, это выражение может ссылаться на любой элемент массива.

Теперь вернемся к решению нашей задачи. Десять переменных мы завели, но теперь в них надо прочитать десять чисел. Это может быть выполнено с помощью оператора:

```
for  i:=1  to  10  do  Read(xx[i]);
```

Оператор будет выполняться так: сначала переменной i будет присвоено значение 1, и будет выполнен оператор Read (xx[i]) (т. е. будет прочитано число в переменную xx[1]); затем значение переменной i будет увеличено на 1 и снова будет выполнен оператор Read (xx[i]); и так будет проделано 10 раз, т. е. оператор Read (xx[i]) будет выполнен 10 раз при значениях i, равных 1, 2, ..., 10. Таким образом, во все

10 переменных xx [1], xx [2], ..., xx [10] будут прочитаны числа.

В общем случае оператор **for** (называемый оператором цикла) имеет вид:

for *имя-переменной*:=*выражение1* to *выражение2* do *оператор;*

Слова **for**, **to** и **do** являются зарезервированными. Переменная, задаваемая *именем-переменной*, называется *переменной цикла* или *параметром цикла*. Значение *выражения1* называется *начальным значением параметра цикла*, а значение *выражения2* — *конечным значением параметра цикла*. *Оператор*, стоящий после зарезервированного слова **do**, называется *телом цикла*.

Оператор цикла **for** выполняется:

Вычисляются значения *выражения1* и *выражения2*.

Вычисленные значения *выражения1* и *выражения2* сравниваются: если значение *выражения1* больше значения *выражения2*, то выполнение оператора **for** на этом заканчивается.

Иначе оператор выполняется (*выражение2* — *выражение1*) +1 раз; при этом перед первым выполнением тела цикла переменной *имя-переменной* присваивается значение *выражения1*, а перед каждым последующим выполнением тела цикла значение переменной цикла увеличивается на 1.

Если в теле цикла нужно использовать не один оператор, а несколько, следует использовать составной оператор.

В рассмотренном операторе **for** переменная цикла в процессе повторений тела цикла увеличивается от начального значения до конечного. Имеется другой вариант оператора **for**, в котором переменная цикла уменьшается (опять же от начального значения до конечного). Этот вариант оператора **for** имеет вид:

for *имя-переменной*:=*выражение1* downto *выражение2* do *оператор;*

В этом варианте тело цикла не будет исполняться ни одного раза, если значение *выражения1* меньше значения *выражения2*; иначе тело цикла будет исполнено (*выражение2* — *выражение1*) +1 раз; перед первым выполнением цикла переменной цикла будет присвоено значение *выражения1*, а перед каждым последующим оно будет уменьшаться на 1.

Заметьте, что количество выполнений тела цикла определяется до первого выполнения тела цикла, и поэтому если в «*выражении2*» используются переменные, которые изменяются в теле цикла, это никак не повлияет на количество выполнений тела цикла. Этим Паскаль отличается от других языков программирования. Например, в следующей последовательности операторов:

```
n:=10;
for  i:=1  to  n
do  begin  Write(i);  n:=100;  end;
```

тело цикла будет выполняться 10 раз, хотя уже после первого выполнения тела цикла конечное значение параметра цикла станет равным 100.

Вернемся ко второй части задачи — поиску максимального числа. Для решения ее нам опять понадобится оператор цикла. Действовать мы будем следующим образом. В переменную xmax, в которой в конце концов окажется максимальный элемент, мы занесем первый элемент массива xx — xx [1]. Затем в цикле будем по очереди сравнивать остальные элементы массива — xx [2], ..., xx [10] — с xmax. Если очередной элемент окажется больше xmax, значение xmax мы заменим на значение этого элемента и продолжим просмотр массива. К концу просмотра в xmax останется максимальный элемент. Таким образом, окончательно программа будет выглядеть так:


```
var  xx   : array [1..10] of real;
     xmax : real;
     i    : integer;
begin
  for  i:=1  to  10  do  Read(xx[i]);
  xmax := xx[1];
  for  i:= 2  to  10
  do  if  xmax < xx[i]
      then  xmax := xx[i];
end.
```

А теперь настало время поговорить о константах. Наша программа обрабатывает 10 чисел. Если нам потребуется обработать 20 чисел, то нам нужно будет в трех местах заменить число 10 на число 20. Если бы наша программа была больше, то, скорее всего, такую замену пришлось бы делать большее число раз. Мало того, что это неудобно, к тому же есть вероятность, что при замене будет пропущена какая-нибудь запись, и мы получим неправильную программу. Чтобы устранить этот недостаток, мы можем завести в программе константу:

```
const  n = 10;
var  xx   : array [1..n] of real;
     xmax : real;
     i    : integer;
begin
  for  i:=1  to  n  do  Read(xx[i]);
  xmax := xx[1];
  for  i:= 2  to  n
  do  if  xmax < xx[i]
      then  xmax := xx[i];
end.
```

Теперь, чтобы изменить количество обрабатываемых программой чисел, достаточно внести изменение только в первую строку.

В общем случае описание константы имеет вид:

const *имя-константы=выражение;*

const — это зарезервированное слово, которым отмечается начало описания константы. Если нужно описать подряд несколько констант, то слово const достаточно поставить только перед первым описанием константы (но можно ставить и перед каждым). *Имя-константы* — это имя, которым называют константу. Далее в программе его можно использовать везде, должно было бы стоять *выражение*. *Выражение* задает значение константы. Это выражение должно быть константным, т. е. должно вычисляться на этапе компиляции; для этого оно не должно содержать переменных.

Константы уместно использовать для обозначения тех величин, которые являются постоянными для данного варианта программы, но могут меняться от варианта к варианту. Также удобно задавать константами величины, которые вы собираетесь подобрать в период отладки программы — например, позиции сообщений на экране или цвета.

Бывает так, что значения одних констант связаны с другими. Например, нужно задать положение какого-то прямоугольного окна на экране и центральную его позицию. Тогда подойдет описание констант вроде:

```
const  x1 = 10;  y1 = 3;              { левый верхний угол окна }
       x2 = 40;  y2 = 10;             { правый нижний угол окна }
       x_c = x1 + ((x2-x1) div 2);    {                   центр }
       y_c = y1 + ((y2-y1) div 2);    {                   центр }
```

Среди массивов выделяется один специальный тип — строки. Переменная строкового типа описывается так:

```
var s:string ["константное выражение"]
```

string — зарезервированное слово (в отличие от, например, integer — предопределенного имени; но пока не обращайте внимания на это отличие). *Константное-выражение* (чаще всего это просто число) задает максимальную длину строки и должно лежать в интервале от 1 до 255. Переменная, описанная как:

```
var  s : string [ n ];
```

во многих отношениях похожа на переменную, описанную как:

```
var s : array [ 0..~n~ ] of char;
```

Например, в обоих случаях можно выбрать i-тый элемент с помощью конструкции s [i]. Но есть и отличия. Строка трактуется как массив переменной длины (правда, память резервируется на максимально возможный размер). Первый элемент строки — s [0] — трактуется как текущая длина строки; она может лежать в диапазоне от 0 до максимального размера строки (заданного в ее описании). (Внимательный, но не искушенный читатель заметит здесь противоречие: s [0] имеет литерный тип, как же это может определять длину? Ответ здесь такой: действительно, тип s [0] литерный, но каждая литера имеет свой числовой код (который и хранится в литерной переменной), и этот-то код и определяет текущую длину строки.) Следующие элементы — в количестве s [0] — есть последовательность литер, составляющих строку. Идущие далее (до максимального размера строки) элементы не должны нормально использоваться (там, конечно, что-то лежит, но что — неопределено).

Константы строкового типа записываются как последовательности литер, ограниченные с обоих сторон апострофами. Например:

```
'абвгде'
'1 2 3'
```

Строки удобно использовать для ввода, вывода и обработки текстов.

По сравнению с обычными массивами, к строкам применима операция конкатенации. Она обозначается знаком «+»; результат ее получается приписыванием к одной строке другой. Например, если значение строковой переменной s1 есть 'Привет', а значение строковой переменной s2 есть 'Вася', то значением выражения:

```
s1 + ', ' +s2 + '!'
```

будет строка 'Привет, Вася!'.

Существует также набор процедур и функций, позволяющих выделить в строке подстроку, удалить подстроку из строки и т. п. Полный перечень этих процедур и функций вы найдете в документации.

Мы научились описывать переменные сложных типов — массивы и строки. Но тот способ, которым мы пока располагаем, имеет очевидное неудобство: если нам в разных местах программы потребуется описать несколько переменных одного и того же сложного типа, нам придется многократно повторять одну и ту же длинную запись. Вместо этого мы можем завести новое имя типа, связав с ним длинное описание типа, и использовать далее везде это имя вместо описания. Это достигается с помощью

«описания типа». Описание типа имеет вид:

type *имя-типа=описание-типа;*

type — это зарезервированное слово, отмечающее начало описания типа. *Имя типа* должно быть новым именем (идентификатором), нигде ранее не использовавшимся. *Описание-типа* мы можем пока изготовить двумя способами: как описание массива или как описание строки. Если подряд идут несколько описаний типов, то зарезервированное слово type можно ставить только перед первым описанием.

Примеры использования описаний типов:

```
type  numbers = array [1..n] of real;
      str5    = string[5];
.....
var  xx : numbers;
.....
var  s1, s2, s3 : str5;
.....
var  yy : numbers;
```

Как мы уже отметили, использование описаний типов позволяет сократить программу. Это является удобством, но не необходимостью. Но в ряде случаев в Паскале невозможно обойтись без использования описаний типов. Например, по правилам языка, тип параметра процедуры или функции (о процедурах и функциях — в следующем разделе) может быть задан только именем типа; поэтому, если параметр должен иметь сложный тип, его нужно сначала связать с каким-либо именем типа в описании типа, и затем использовать это имя в заголовке процедуры или функции.

В рассмотренном нами ранее операторе цикла for количество выполнений тела цикла определяется в начале исполнения оператора, до первого выполнения тела цикла. Но часто условие завершения цикла определяется в процессе выполнения тела цикла. В таких случаях следует использовать один из других двух операторов цикла: while или repeat.

Оператор while имеет следующий формат:

```
while   "логическое-выражение"
do  "оператор"
```

Здесь while и do — зарезервированные слова; *оператор*, как и в случае оператора for, называется телом цикла; *логическое-выражение* называется условием цикла.

Оператор while выполняется следующим образом: вначале вычисляется *логическое-выражение;* если его значение есть «истина», то выполняется *оператор*; после этого *логическое-выражение* вычисляется снова, и т. д. Когда очередное вычисление *логического-выражения* дает «ложь», выполнение оператора цикла завершается (в частности, если при первом вычислении *логического-выражения* получится «ложь», тело цикла не будет выполнено ни разу).

Оператор repeat имеет следующий формат:

```
repeat
   "последовательность-операторов"
until  "логическое-выражение"
```

Здесь repeat и until — зарезервированные слова; *последовательность-операторов* называется телом цикла; *логическое-выражение* называется условием цикла.

Оператор repeat выполняется следующим способом: сначала выполняется *последовательность-операторов;* затем вычисляется *логическое-выражение*; если его значение есть «истина», то на этом выполнение оператора repeat завершается; если же его значение есть «ложь», то снова выполняется *последовательность-операторов,* снова вычисляется *логическое-выражение* и т. д.

Операторы while и repeat очень похожи друг на друга; однако следует обратить внимание на два отличия:

В операторе while условие цикла проверяется *перед* выполнением тела цикла, а в операторе repeat — после; поэтому тело цикла в операторе repeat всегда будет выполнено по крайней мере один раз.

Оператор while выполняется до тех пор, пока условие цикла остается истинным; как только оно становится ложным, выполнение оператора while завершается. Оператор repeat выполняется до тех пор, пока условие цикла остается ложным; как только оно становится истинным, выполнение оператора repeat завершается.

Обратите внимание на одно существенное отличие оператора for от операторов while и repeat; в операторе for значения выражений, определяющие количество исполнений тела цикла, вычисляются один раз, в начале выполнения оператора цикла; в операторах while и repeat *логическое-выражение* вычисляется снова после каждого выполнения тела цикла.

Вы, вероятно, заметили еще одно отличие: в операторах for и while тело цикла состоит из одного оператора, а в операторе for — из последовательности операторов. Причина этого проста. В операторе repeat есть естественные «скобки» — зарезервированные слова repeat и until — которые ограничивают тело цикла. В операторах for и while таких «скобок» нет; также нет их и в операторе if. В этом, пожалуй, Паскаль отстает от более новых языков, в которые специально добавлены такие «скобки».

```
    else
       "последовательность-операторов"
    fi ;

a оператор  "while":

    while "условие"
    do "последовательность-операторов"
    od ;
```

Новые зарезервированные слова — **fi** и **od** — выполняют здесь роль правых скобок, позволяющих определить конец последовательности операторов.

Приведем простые примеры использования операторов **while** и **repeat**.

Пусть требуется найти позицию первого пробела в строковой переменной s (Точнее: переменной целого типа i требуется присвоить индекс первого пробела в строковой переменной s, если он там имеется, или длину строки, если строка не содержит пробела.) Это выполняет следующий фрагмент программы:


```
i := 1;
while  (i<=Length(s)) and (s[i]<>' ')
do  i := i+1;
```

Функция Length имеет в качестве параметра выражение строкового типа и возвращает в качестве результата текущую длину строки.

Пусть требуется дождаться ввода с клавиатуры возврата каретки (все остальные нажатия клавиш до нажатия возврата каретки должны быть проигнорированы). Это выполняется следующим оператором: (13 —

```
repeat  until  ReadKey = #13;
```

это код возврата каретки). Обратите (13 — это код возврата каретки). Обратите внимание — тело цикла здесь пустое! Действительно, нам ничего не нужно делать — только повторять вызовы функции **ReadKey** до тех пор, пока она не возвратит # 13.

Однако бывают задачи, для решения которых недостаточно рассмотренного набора операторов (по крайней мере, недостаточно для естественного решения задачи). Это задачи, в которых условие окончания цикла определяется внутри тела цикла. В качестве примера такой задачи рассмотрим следующую.

Нам требуется упорядочить по убыванию массив вещественных чисел, т. е. поставить на первое место самое большое число, на второе — следующее по величине и т. д. Мы будем решать эту задачу таким способом: сначала найдем в массиве самый большой элемент и поставим его на первое место; затем найдем самый большой в остатке массива элемент и поставим его на второе место и т. д. (Это не самый эффективный метод решения поставленной задачи; если вам потребуется быстрое упорядочивание, воспользуйтесь другим, более быстрым, алгоритмом. Мы рассматриваем здесь такой алгоритм потому, что он прост и удобен для демонстрации интересующей нас проблемы.) Следующий фрагмент программы реализует сформулированный алгоритм:

```
const  n = 10;
var  xx   : array [1..n] of real;
     tmp  : real;
     i, j : integer;
. . . . .
begin
. . . . .
  for  i:=1  to  n-1
  do  for  j:=i+1  to  n
      do  if  xx[j] > xx[i]
          then  begin
                  tmp:=xx[i];
                  xx[i]:=xx[j];
                  xx[j]:=tmp;
                end;
. . . . .
end.
```

Пока что все в порядке: нам удалось справиться с поставленной задачей известными нам средствами. Но давайте потребуем, чтобы работа была прекращена (и массив оставлен в недоупорядоченном виде) как только будет нажата какая-нибудь клавиша на клавиатуре. Тут нам хотелось бы записать циклы как-то вроде:

```
for  i:=1  to  n-1
do  for  j:=i+1  to  n
    do  if  KeyPressed
        then  { выйти из циклов }
        else  if  xx[j] > xx[i]
              then  begin
                      tmp:=xx[i];
                      xx[i]:=xx[j];
                      xx[j]:=tmp;
                    end;
```

Для того чтобы так и сделать, нам потребуется *оператор перехода* **goto**. Оператор **goto** имеет формат:

goto *метка;*

goto — зарезервированное слово. *Метка* — это либо целое число без знака, либо идентификатор. Любая метка, кроме (возможно, многочисленных) использований в операторах **goto**, должна появиться еще в двух местах:

В описании меток, располагаемом в разделе описаний (там же, где располагаются описания констант, переменных и типов). Описание меток состоит из зарезервированного слова **label** и следующего за ним списка меток (в котором метки отделяются друг от друга запятыми). Например:

```
label  11, StartOfWork, 999;
```

Перед каким-либо оператором в разделе операторов. Здесь метка отделяется от следующего за ней оператора двоеточием. Перед одним оператором может находиться несколько меток; в этом случае после каждой метки должно стоять двоеточие. В таком случае говорят, что данный оператор *помечен* меткой.

Действие оператора **goto** состоит в том, что после него будет выполняться не следующий по порядку в программе оператор, а оператор, помеченный указанной в операторе **goto** меткой (иначе говорят, что оператор **goto** передает управление на помеченный меткой оператор).

Окончательно программа нашего примера будет выглядеть так:

```
const  n = 10;
var  xx   : array [1..n] of real;
     tmp  : real;
     i, j : integer;
label  EndOfLoops;
. . . . .
begin
. . . . .
  for  i:=1  to  n-1
  do  for  j:=i+1  to  n
      do  if  KeyPressed
          then  goto  EndOfLoops
          else  if  xx[j] > xx[i]
                then  begin
                        tmp:=xx[i];
                        xx[i]:=xx[j];
                        xx[j]:=tmp;
                      end;
EndOfLoops:  . . .
. . . . .
end.
```

Оператор **goto** является мощным инструментом программирования. Машинная команда, являющаяся аналогом оператора **goto**, является совершенно необходимой: можно отказаться от команды умножения и выполнять умножение с помощью процедур, использующих только команды сложения (кстати, так и происходит в «Корвете»: микропроцессор Intel 8080 не имеет команды умножения); можно отказаться практически от каждой из остальных команд и сохранить возможность (пусть и с большим трудом) писать программы — но от команды перехода отказаться нельзя. В старых языках программирования — например, в Фортране — также нельзя обойтись без оператора **goto**.

Но чрезмерное увлечение оператором **goto** приводит к тому, что программа становится запутанной и, как следствие, трудноотлаживаемой и малопонятной.

В моей практике был такой случай. Меня попросили найти ошибку в одной программе. Программа была написана на языке PL/1 и занимала чуть больше печатной страницы. Я бодро взялся за дело. Уже в третьей строке я обнаружил переход на пятую с конца строку. Это несколько смутило меня, но я продолжал расследование. В пятой с конца строке оказался переход на пятую с начала строку, а там — снова переход куда-то в середину программы. Куда был следующий переход, я не стал смотреть: я заявил, что мне легче написать новую программу, чем искать ошибку в имеющейся. (В конечном счете мне и пришлось написать новую программу; в ней я ни разу не использовал **goto**, и заработала она сразу.)

Один из принципов структурного программирования состоит в отказе от использования **goto**. Правда, для того чтобы необходимость использования **goto** полностью отпала, требуется расширение набора управляющих конструкций языка. Паскаль частично удовлетворяет этим требованиям. С одной стороны, набор операторов в нем достаточен для того, чтобы любая программа могла быть написана без использования **goto**. Одиако иногда программа от этого становится только более непонятной. Например, наша программа могла бы быть так переписана без использования **goto**.


```
const  n = 10;
var  xx   : array [1..n] of real;
     tmp  : real;
     i, j : integer;
     work_flag : boolean;
. . . . .
begin
. . . . .
  work_flag := true;
  i := 1;
  while  work_flag and (i <= n-1)
  do  begin
        j := i+1;
        while  work_flag and (j <= n)
        do  if  KeyPressed
            then  work_flag := false
            else  begin
                    if  xx[j] > xx[i]
                    then  begin
                            tmp:=xx[i];
                            xx[i]:=xx[j];
                            xx[j]:=tmp;
                          end;
                    j := j+1;
                  end;
        i := i+1;
      end;
. . . . .
end.
```

Вряд ли этот текст покажется вам более понятным, чем его предыдущий вариант.

Но все-таки использование **goto** в Паскале требуется относительно редко. И чтобы отбить излишнюю охоту к использованию **goto**, разработчики языка специально постарались сделать такое использование не очень удобным. Мы, со своей стороны, рекомен-

дуем вам использовать goto только в следующих ситуациях (в них, кстати, использование goto не уменьшает понятность программы):

Для выхода из оператора цикла. В этом случае метка, на которую передается управление, должна находиться непосредственно после оператора, из которого производится выход.

Для выхода из процедуры (о том, что такое процедура, см. следующий раздел). В этом случае метка, на которую передается управление, должна находиться или непосредственно перед завершающим процедуру 'end', или перед группой операторов, завершающих выполнение процедуры (в этой группе операторов не должно содержаться переходов на начало процедуры).

Для перехода на начало блока в случае, если в процессе выполнения блока обнаруживается необходимость начать все действия сначала (например, при обработке введенной человеком команды: если в команде обнаруживается ошибка, нужно выдать диагностику и повторить ввод команды сначала).

Возможно, вы найдете и другие случаи, в которых уместно использование goto. Мы бы хотели, чтобы приведенные выше правила сделали бы вам более понятным дух, а не букву структурного программирования.

Продолжение следует

А. ДЕВЯТОВ

Программирование 9-игольчатых матричных принтеров семейства Epson

Таблицы символов

Все кодировки символов, принятые в нашей стране, имеют общую часть с кодами 0—127. Эта часть полностью заимствована из стандарта ASCII (American Standart Code for Information Interchange).

Различия существуют в стандартизации кириллицы и псевдографических символов, используемых для построения табличных рамок (коды 128—255).

В таблицах 1—6 приводятся стандарт ASCII (коды 0—127), а также наиболее распространенные стандарты кодировки кириллицы и символов псевдографики (коды 128—255).

Таблица 1

Код	00	10	20	30	40	50	60	70
0	NUL		SP	0	a	P	′	p
1		DC1	!	1	A	Q	a	q
2		DC2	″	2	B	R	b	r
3		DC3	#	3	C	S	c	s
4		DC4	$	4	D	T	d	t
5			%	5	E	U	e	u
6			&	6	F	V	f	v
7	BEL		′	7	G	W	g	w
8	BS	CAN	(	8	H	X	h	x
9	HT	EM	)	9	I	Y	i	y
A	LF		*	:	J	Z	j	z
B	VT	ESC	+	;	K	[	k	{
C	FF		,	<	L	\	l	|
D	CR		—	=	M	]	m	}
E	SO		.	>	N	â	n	~
F	SI		/	?	O	—	o	

Стандарт ASCII
(общая часть всех кодировок)

Таблица 2

Код	80	90	AO	BC	CO	DO	EO	FO
0					ю	п	Ю	П
1					а	я	А	Я
2					б	р	Б	Р
3					ц	с	Ц	С
4					д	т	Д	Т
5					е	у	Е	У
6					ф	ж	Ф	Ж
7					г	в	Г	В
8					х	ь	Х	Ь
9					и	ы	И	Ы
A					й	з	И	З
B					к	ш	К	Ш
C					л	э	Л	Э
D					м	щ	М	Щ
E					н	ч	Н	Ч
F					о	ъ	О	Ъ

Код кириллицы КОИ-8

*Окончание. Начало см.: ИНФО. 1992 № 3—4.

Таблица 3

Код	80	90	A0	B0	C0	D0	E0	F0
0				А	Р	а	р	Ё
1				Б	С	б	с	ё
2				В	Т	в	т	
3				Г	У	г	у	
4				Д	Ф	д	ф	
5				Е	Х	е	х	
6				Ж	Ц	ж	ц	
7				З	Ч	з	ч	
8				И	Ш	и	ш	
9				И	Щ	й	щ	
A				К	Ъ	к	ъ	
B				Л	Ы	л	ы	
C				М	Ь	м	ь	
D				Н	Э	н	э	
E				О	Ю	о	ю	
F				П	Я	п	я	

Основная кодировка ГОСТ кириллицы

Таблица 4

Код	80	90	A0	B0	C0	D0	E0	F0
0	А	Р	а				р	Ё
1	Б	С	б				с	ё
2	В	Т	в				т	
3	Г	У	г				у	
4	Д	Ф	д				ф	
5	Е	Х	е				х	
6	Ж	Ц	ж				ц	
7	З	Ч	з				ч	
8	И	Ш	и				ш	
9	Й	Щ	й				щ	
A	К	Ъ	к				ъ	
B	Л	Ы	л				ы	
C	М	Ь	м				ь	
D	Н	Э	н				э	
E	О	Ю	о				ю	
F	П	Я	п				я	

Альтернативная кодировка ГОСТ кириллицы

Таблица 5

Код	80	90	A0	B0	C0	D0	E0	F0
0				О	Ю		о	ю
1			А	П	Я	а	п	я
2			Б	Р		б	р	
3			В	С		в	с	
4			Г	Т		г	т	
5			Д	У		д	у	
6			Е	Ф		е	ф	
7			Ё	Х		Ё	Х	
8			Ж	Ц		Ж	Ц	
9			З	Ч		з	ч	
A			И	Ш		и	щ	
B			Й	Щ		й	щ	
C			К	Ъ		к	ъ	
D			Л	Ы		л	ы	
E			М	Ь		м	ь	
F			Н	Э		н	э	

Кодировка кириллицы в принтере «Ravi-8010M»

Таблица 6

	1	2	3		1	2	3
А	128	176	225	а	160	208	193
Б	129	177	226	б	161	209	194
В	130	178	247	в	162	210	215
Г	131	179	231	г	163	211	199
Д	132	190	228	д	164	212	196
Е	133	181	229	е	165	213	197
Ё	240	240	нет	ё	241	241	нет
Ж	134	182	246	ж	166	214	214
З	135	183	250	з	167	215	218
И	136	184	233	и	168	216	201
Й	137	185	234	й	169	217	202
К	138	186	235	к	170	218	203
Л	139	187	236	л	171	219	204
М	140	188	237	м	172	220	205
Н	141	189	238	н	173	221	206
О	142	190	239	о	174	222	207
П	143	191	240	п	175	223	208
Р	144	192	242	р	224	224	210
С	145	193	243	с	225	225	211
Т	146	194	244	т	226	226	212
У	147	195	245	у	227	227	213
Ф	148	196	230	ф	228	228	198
Х	149	197	232	х	229	229	200
Ц	150	198	227	ц	230	230	195
Ч	151	199	254	ч	231	231	222
Ш	152	200	251	ш	232	232	219
Щ	153	201	253	щ	233	233	221
Ъ	154	202	255	ъ	234	234	223
Ы	155	203	249	ы	235	235	217
Ь	156	204	248	ь	236	236	216
Э	157	205	252	э	237	237	220
Ю	158	206	224	ю	238	238	192
Я	159	207	241	я	239	239	209

Объединенная таблица кодировок кириллицы
1 — альтернативная кодировка ГОСТ;
2 — основная кодировка ГОСТ;
3 — кодировка КОИ-8.

БЮРО ИНТЕЛЛЕКТУАЛЬНЫХ ТЕХНОЛОГИЙ

Ф-БИТ

Бюро Интеллектуальных Технологий Ф-Бит занимается разработкой и реализацией программных и программно-аппаратных средств для УКНЦ и IBM-совместимых компьютеров. Предлагаемый набор продуктов охватывает многие области применения компьютерной техники - от образования и бухгалтерского учета до мощных пользовательских сред ведущих фирм мира (Microsoft, Novell, Borland, ...).

- ✓ разработка оригинальных программных учебных средств
- ✓ разработка методик предметных уроков и курсов ОИВТ
- ✓ разработка аппаратных комплексов
- ✓ издательская деятельность
- ✓ поставка и обслуживание компьютерных классов
- ✓ обучение и методическое сопровождение
- ✓ исследования в области чистой математики
- ✓ сотрудничество с ведущими разработчиками программных и аппаратных средств

НАШ АДРЕС

103498 Москва, К-498, а/я 164

ЗВОНИТЕ НАМ ПО ТЕЛЕФОНАМ

535-22-22
534-48-31 (модем)
535-26-05 (факс)

Ф-Бит распространяет следующие продукты для УКНЦ:

Игровые пакеты

Игрушка
Игрушка 2
Игрушка 3
Игрушка 4

Аппаратные средства

электронный диск
программно-аппаратный комплекс AMJ-T
программно-аппаратный комплекс Механика
программно-аппаратный комплекс Секретарь
телефонный сервер
музыкальная приставка

Инструментальные программы

язык программирования Mit-Logo
пакет UKGRAPH
компилятор языка Си
операционная система реального времени РУДОС

Учебные средства для начальной школы

Роботландия
Мозаика
Matris
Сетевое лото

Учебные средства по ОИВТ

ИнфоМир
Информатика-91
Сократ

Учебные средства для предметных уроков

Математика-91

Прикладные программы

текстовый редактор Writer
графический редактор Квант
программа распечатки изображений UG Print
музыкальный редактор Моцарт
графический редактор COLOC
программа печати документов UK Print
командная оболочка PAF Commander
утилита Tank Formatter
система подготовки данных Мини-Канцелярия
система подготовки данных Мини-Бухгалтерия
автоматизированная система Склад & Реализация
игровой клавиатурный тренажер Touch Typing

Сетевые программные средства

сетевой монитор BAM
интерфейс пользователя локальной сети

Каталог ежемесячно пополняется новыми программами

☎ 535-22-22 ☎ 534-48-31

В. КОВАЛЬ

Разработка Grab-программ на Паскале

Тем, кто работает с вычислительной техникой, нередко приходится сталкиваться с необходимостью повторного использования текстовой информации, выводимой на экран монитора. Для этого бывает недостаточно вывода содержимого экрана на принтер, поскольку очень часто необходимо использовать полученную информацию для вставки в тексты, описания, статьи и т. п. Для решения подобного рода задач предлагается резидентная программа, которая позволяет пользователю получить точную копию экрана монитора в виде обычного текстового файла, который может использоваться по усмотрению.

Такие программы существуют в основном для графических сред. Они адаптированы под формат записи определенного графического редактора и могут использоваться только в его пределах. Но такая постановка вопроса не всегда удовлетворяет пользователей. Поэтому автором данной статьи разработана небольшая Grab-программа, позволяющая производить описанные выше действия применительно к текстовым задачам.

Вообще разработка подобных программ является очень трудоемким занятием. Существует мнение, что для таких целей, как нельзя более подходит язык Ассемблера. Но в последнее время в такой популярной системе программирования как Turbo Pascal появились средства, с помощью которых можно разрабатывать резидентные приложения на языке Паскаль. Естественно, что они гораздо компактнее по тексту и более просты для восприятия. Для примера можно сказать, что подобного рода программа на языке Ассемблера занимает несколько тысяч строк.

Сделав некоторые изменения в этой программе, ее можно использовать для сохранения и дальнейшего использования в программах пользователя графических изображений. Для этого необходимо заменить процедуру MainProc на следующую:

```
{$S-,R-,V-,B-}
{$M $400,0,0}    {1 Kb for stack}
programm Grab;
uses Dos;


   const
  IntFO     : Word     = $CF90;
  OldKbd    : Pointner = Nil;
  OldIntFO  : Pointner = Nil;
  Count     : Integrer = 0;
  Maska     : Byte     = $03;

var
  KeyFlag: Byte absolute $0:$417;
       {Абсолютный адрес флагов
      клавиатуры в области BIOS}
  TsrSS, TsrSP:Word;
    { для сохранения SS и SP
      нашей программы }
  OldSS, OldSp:Word;
    { для сохранения SS и SP
      прерванной программы }

  function Check : Boolean;
begin
  GetIntVec($FO,OldIntFO);
  Check := (OldIntFO<>Nil)
and(Word(OldIntFO^)=IntFO)
end;

procedure InitData;
       {начальные установки}
begin
 TsrSS := SSeg;
 TsrSP := SPtr;
 GetIntVec ( $09, OldKbdInt);
 GetIntVec ( $FO, OldIntFO );
end;


procedure CLI; assembler;
asm
  CLI
end;

procedure STI; assembler;
asm
  STI
end;


procedure CallOldInt09;
{эмуляция вызова  старого
прерывания int 09h}
inline($9C/   {pushf; сохранение
                    флагов}
       $3E/   {DS   ; переход на
               адресацию DS  }
       $FF/$1E/OldKbdInt);
         {Call far OldKbdInt }

procedure SaveStackRegs;

 { сохраняет значения стековых
    регистров}
 inline($3E/
        $8C/$16/OldSS/
{mov OldSS,SS ; сегмент стека }
        $3E/
        $89/$26/OldSP/
{mov OldSP,SP ;указатель стека }
        $3E/
        $8E/$16/N=TsrSS/
{mov SS,TsrSS ;установка}
$3E/
$8B/$26/TsrSP);
{mov SP,TsrSP ;          }


procedure RestoreStackRegs;
{востанавливает значение стековых
             регистров}
```

```
inline($3E/
       $8E/$16/OldSS/
     "mov SS,OldSS}
       $3E/
       $8B/$26/OldSp);
     "mov SP,OldSP}

procedure MainProc;
"процедура копирования
 содержимого текстового экрана}
var S  : String;4¿;
   i,j : Integrer;
     f : Text;
begin
   CLI;SaveStackRegs; STI;
"запомнить стековые регистры
  прерванной программы}

 Str (Count,S );
 Assign(f,'file'+S+'.txt');
 Rewrite(f);
 i := 0; j : =1;
 While i<=50*80 do "25*2 строк,
          "80 символов/строку }
begin
   Write(f,Chr(Mem;$B800: i¿));

if(j mod 80) = 0 then
          begin
           Writeln(f);
            j : =0
    end;
 i :=i+2; j : =j+1
  end;
Close(f);
Count := Count+1;
CLI; RestoreStackRegs; STI
 "востановить стековые регистры
   прерванной программы}

end;

procedure
MyKbdInt(Flags,CS,IP,AX,BX,
     CX,DX,SI,DI,DS,ES,BP,:Word);
interrupt;
begin
  CLI;
 Call01Int09;
 "выполняем старую процедуру
 обработки INT 09H }
if ((KeyFlag)and(Maska)=Maska)
  then MainProc;
 STI
end;
begin
 SwapVectors;
  if Check then
     begin
  Writeln('Grab Already Loaded.'+
'Press LeftShift+RightShift for
  activate');
 Halt(0)
  end;
```

```
InitData;
Writeln('Press LeftShift+
 RightShift for activate
 Grab Programm');
  SetIntVec($FO, @IntFO);
  SetIntVec($09, @MyKbdInt);
 Keep(0);
end.
```

```
procedure MainProc;
"процедура копирования содержимого
    экрана}
  var S : String;4¿;
      i : Integrer;
      f : file of Byte;
begin
    CLI;SaveStackRegs; STI;
  "запомнить стековые регистры
     прерванной программы}
 Str(Count,S);
 Assign(f,'file'+S+'.pic');
  ReWrite(f);
   i :=1;
     for i:=1 to 16*1024 do
 Write(f,Mem;$B800:i¿);
Close(f);
  Count := Count+1;
 CLI; RestoreStackRegs; STI
  "востановить стековые регистры
    прерванной программы}
end;
```


В этой процедуре основной упор делается на то, что происходит копирование видеобуфера CGA.

Эти программы после компиляции занимают в памяти всего около 6 Кбайт, что дало возможность их применения на учебных ЭВМ типа «ПОИСК».

Хочу также отметить, что выше приведены лишь упрощенные варианты программ с учетом журнальных ограничений. С теми, кто заинтересовался данной темой, автор поделится своими соображениями и предоставит более мощный вариант программы, которая автоматически распознает режим работы адаптера (графический или текстовый) и «снимает» графическое изображение по точкам, что делает программу независимой от типа адаптера.

Работы производились на ПЭВМ «Нейрон» с применением системы Turbo Pascal 6.0 и на учебной ЭВМ «ПОИСК» с применением той же системы в варианте компилятора командной строки.

Доска объявлений

Предлагаю для обмена свыше 100 программ для «Спектрума».
450055, г. Уфа, пр. Октября, д. 136/1, кв. 34. Прошин Антон.

Хочу переписываться с владельцами ПЭВМ «Агат-9», ищу программы ПЭВМ «Apple-11», «Правец 8М», «Агат-7», «BBC».
404120, Волгоградская обл., г. Волжский, пр. Ленина, 119, кв. 172. Сопкин А. Ю.

Хотел бы переписываться и обмениваться программами с владельцами ПК «Партнер-01.01».
658031, Алтайский край, Тальменский р-н, село Озерки, ул. Зеленая, 18. Камнев Ю. А.

Хочу переписываться и обмениваться программами с владельцами ПЭВМ «Вектор-06Ц».
443094, Самара, ул. Стара-Загора, 209, кв. 358. Шубин Г. В.

Просим откликнуться владельцев «Радио-РК86» и помочь в создании компьютерной газеты на кассете. Присылайте свои статьи, заметки, объявления. Мы очень ждем!
450092, Уфа, ул. Авроры, д. 11/2, кв. 24. Баймуратов И. Ю.

Прошу откликнуться любителей, занимающихся компьютером «Коммодор» (Commodore-16, +4 или же VIC-20).
398035, Липецк, ул. Космонавтов, 41/2, кв. 83. Котов В.

Предлагаю обмен программами для УКНЦ. В письмо прошу вложить каталог.
429200, Чувашия, п. Вурнары, а/я 10. Александр Николаевич.

Прошу откликнуться владельцев компьютера «Ириша» (дисковой вариант). Нужно программное обеспечение. Оплату гарантирую.
182150, Псковская обл., Великолукский р-н, пос. Поречье, ул. Советская, 32, кв. 3. Травкин В. А.

Обмениваюсь программами к ПК «Вектор 06Ц». Имею более 100 ROM или MON программ.
242500, Брянская обл., г. Карачев, 6-А, д. 76, кв. 27. Филатов Д. Н.

Хотел бы переписываться и обмениваться программами для ПЭВМ «Агат», «Спектрум». Программирую на Бейсике, Паскале. Имею около 500 различных программ.
470056, Караганда, 17-й микрорайон, 53, кв. 9. Фиронов В.

Хотел бы переписываться и обмениваться программами для ПК IBM RS/2 и «Корвет». Имею обширный банк программ для IBM PS/2. Высылаю каталоги. В письмо прошу вложить каталог.
703012, Узбекистан, Самарканд, ул. Мукими, 8а, Шабанов А. К.

Обменяюсь или приобрету дисковые TR-DOS программы для ZX-Spektrum-128 (48). Интересуют музыкальные программы, обслуживаемые АУ-3-8912 — редакторы, топы, ритм-боксы, а также системные программы, новые игры (90-91 г.) и т. п.
232007, Литовская республика, г. Вильнюс-7, ул. Кривю, 29, кв. 18. Дейчко П. И.

Предлагаю для обмена большой выбор системных и игровых программ для ПК «Поиск».
197198, Санкт-Петербург, ул. Съезжинская, 13, кв. 36. Бессалов А. Н.

Хотел бы обмениваться программами и информацией для ПЭВМ «Вектор-06Ц». Имею большое количество программ. Вышлю каталог.
375033, Армения, г. Ереван, ул. Аветисяна, 80, кв. 3. Тиграпян А. С.

Хочу переписываться с владельцами ПК «Вектор-06Ц». Готов обмениваться программами и информацией. Вышлю каталог.
375001, Ереван, ул. Абовяна, 12, кв. 60. Казарян А. А.

Прошу откликнуться тех, кто сумел подсоединить джойстик к БК-0011. Документацию оплачу.
470072, г. Караганда, Степной-4, д. 7, кв. 53. Воронцов Л.

Ищу программы для БК-0011М. Оплату гарантирую.
446028, Самарская обл., г. Сызрань, ул. Королева, 13, кв. 114. Середа И. А.

Маленькие хитрости

Хочу предложить читателям «МИ» вариант модификации программы на Basice в процессе ее работы. Предложение основывается на работе функциональных клавиш, поэтому им могут воспользоваться при работе на всех тех компьютерах, на которых они есть.

Допустим нужно вставить строки в программу, необходимые для дальнейшей ее работы. Например, строку, содержащую какую-либо функцию, задаваемую пользователем.

Это возможно, используя функциональные клавиши (F1, F2, ..., F10) или (K1, K2, ..., K10). Они могут запоминать 15 символов некоторого текста, включая коды управляющих клавиш. И если в процессе работы программы, будет нажата одна из функциональных клавиш, то закодированный в ней текст попадет в память компьютера аналогично тексту, набираемому вручную на клавиатуре. Но это произойдет не сразу, а только после остановки работы программы, что можно выгодно использовать.

К примеру, нам надо ввести функцию, график которой надо построить. Это делает маленькая подпрограмма.

```
10 KEY5,CHR$(13)+CHR$(13)+CHR$(13)
20 INPUT "Чему равно f(x)? (после за-
                    писи нажмите F5)";U$
30 LOCATE 0,6
40 PRINT "80 DEF FN f(x)="+U$
50 PRINT "GOTO";N
60 LOCATE 0,5
70 END
```

При обращении к этой подпрограмме в строке 10 произойдет запись в функциональную клавишу F5 3-х кодов клавиши ввода, а в строке 20 переменной U присвоится заданная функция. После нажатия клавиши F5 (K5) один код клавиши ввода завершит выполнение оператора INPUT, два остальных — выполнятся только после остановки программы.

Операторы в строках 30, 40 выводят на экран (начиная с позиции 0,4) строку, которая затем введется в программу.

После выполнения оператора PRINT в строке 50 на экране компьютера появляется команда перехода к нужной строке (например, 999, если N-999) после окончания работы рассматриваемой подпрограммы ввода функции.

Строка 60 отправит курсор на позицию выше для того, чтобы обойти «Ок», который компьютер выведет на экран после остановки программы.

Таким образом, после выполнения строк 20—50 экран компьютера будет выглядеть следующим образом:

```
 0123456789...
0Чему равно f(x)? (после записи нажмите F5)?  sin(x)
1
2
3
4
5
6 80 DEF FN f(x)=sin(x)
7 GOTO 999
.
.
.
```

После выполнения строки 70 произойдет временная остановка программы, в строке 5 экрана появится «Ок», а в строке 6 и 7 будут выведены коды клавиши ввода (13). При этом строка с номером 80 будет вставлена в программу, а команда GOTO 999 просто будет выполнена.

Описанный выше прием позволяет вести до 13 строк в программу во время ее выполнения.

Если программа не очень длинная, описанные выше действия выполняются прак-

тически мгновенно и остановка незаметна.

Следует также заметить, что опытный программист может без труда замаскировать выполняемые действия изменением цвета выводимого на экран текста.

И. Поболков,
ученик 11 кл.
СШ 5, г. Борисоглебск.

Такой же, только без крыльев (Полудетективная статья)

Ни для кого не секрет, что ПЭВМ «Агат» имеет более скудное программное обеспечение, чем его IBM-совместимые коллеги. Поэтому подбор программного средства для решения конкретной задачи на «Агате» представляет довольно сложную задачу, особенно для начинающего пользователя, «коллекция» программ которого еще не очень богата. Программист постоянно встречается с программами, которые могут сделать много полезного, но не умеют какую-нибудь мелочь. Но для конкретной задачи эта «мелочь» является решающей, и не позволяет полноценно применять это программное средство. Пользователь оказывается в положении никулинского героя, вопрошающего: «А нет ли у вас такого же, только без крыльев?». Такая ситуация легко объясняется тем, что разработчик программного средства просто не в силах предусмотреть все и «объять необъятное». Однако это объяснение мало утешает.

Можно, конечно, пользоваться несколькими системными средствами для решения одной задачи, однако время, затраченное на переход от одной системы к другой резко снижает КПД программиста, и кроме того (и это самое главное), какую-нибудь операцию может не позволить произвести ни одна из имеющихся в наличии программ.

Думаю, вы уже поняли, к чему этот разговор. Если вы уверены в своих силах, то можете попытаться встроить свой собственный драйвер даже в программу, составленную не вами. Как правило, программные средства написаны на ассемблере, и достать их исходный модуль обычному пользователю не представляется возможным. Однако если вы владеете ассемблером, то сможете вставить «заплату» в готовую программу. Данная статья поможет вам в этом.

Подключение новых блоков — довольно сложная, но в то же время увлекательная процедура, особенно для любителей... детективных романов. Программист, выступающий в роли следователя, должен пройти практически все этапы следствия (правда, не всегда в том же порядке), и раскрыть «преступление». Непродуманные действия программиста, как и следователя, могут закончить дело «глухарем». Однако задача практически всегда имеет множество решений, и достаточно найти хотя бы одно из них. Итак, приступим.

1. Определение мотивов преступления

Сначала попытаемся четко сформулировать, что именно нас не устраивает в имеющейся программе. Например, надо изменить клавиши управления, установить другой цвет вывода, обеспечить работу программы с другим типом принтера, и т. д. Понять, что надо сделать — задача, по важности приравниваемая к самому решению.

2. Составление фоторобота преступника

Формулировка задачи должна помочь вам понять, в какой именно блок программы необходимо вмешательство. Например, изменение рабочих клавиш лучше всего произвести в блоке опроса клавиатуры и анализа кодов: изменение цвета — в блоке, выводящем эту информацию; переход к другому принтеру — или непосредственно в подпрограмме печати байта, или (если сообщений мало) — в самих сообщениях.

Определив блок, подумайте:

а) как бы вы написали этот блок?

б) Без каких частей алгоритма разработчик программы никак не мог обойтись?

Например, блок работы с клавиатурой может выглядеть так: опрос клавиатуры ⟶ последовательное сравнение полученного кода с эталонными ⟶ условный переход при совпадении; вывод строки на экран или принтер — передача адреса начала сообщения (или смещения относительно начала другой строки) подпрограмме вывода; подпрограмма опроса пультов: включение одновибратора ⟶ циклический опрос состояний пультов.

3. Поиск орудий преступления

А теперь вам предстоит ответить на вопрос: какие последовательности байтов обязательно встретятся в изменяемом блоке. На первый взгляд, вопрос кажется аб-

сурдным: ведь это зависит от конкретной реализации конкретного алгоритма. Но если вы четко представляете себе связь между командой ассемблера и ее представлением в машинных кодах, то, думаю, сможете ответить на поставленный вопрос.

Например, если программа пользуется подпрограммами вывода символа на экран COUT или COUT1 (СМ Бейсика, адреса $FDDC и $FDD7), то в ней встретятся комбинации байтов $DC, $FD или $D7, $FD. Если изменяется цвет или режим вывода (то есть байт атрибута текста), то почти всегда происходит обращение к ячейке $32, и в программе тоже встретится байт $32.

И даже если вы не смогли придумать характерных байтов для самого блока, можно попробовать «танцевать от печки», хотя это и дольше. Например, вы «ничего такого» не знаете о блоке программы, знаете только, что в работающей программе он вызывается клавишей «S». В этом случае в программе после опроса клавиатуры произойдет сравнение с шаблонными значениями, в том числе и с «S». А этот шаблон наиболее вероятно отобразится в памяти байтом $53 или $D3. Аналогично, если вам известен какой-либо числовой параметр (например, число «жизней» в игровой программе), то соответствующий байт с большой вероятностью встретится в программе.

Не следует, однако, забывать о том, что существуют индексированные адресации. Например, если программа опрашивает пульты 0 и 1, то комбинации байтов $65, $C0 в программе может и не быть, как нет и прямого обращения в ячейке $C065. Обычно пульты опрашиваются обращением к $C064, X, и в X помещается номер опрашиваемого пульта. Конечно, теоретически можно и клавиатуру опросить через ячейку $BF80, X при X-80, но практической пользы это не дает, и поэтому маловероятно. А вот числовой параметр вполне может отличаться от своего «отражения» в памяти с гораздо большей вероятностью. Например, если машина N раз выполняет какое-либо действие, то соответствующий цикл может быть запущен от 1 до N, от N до 1, а может и от 0 до N—1. И к тому же конец цикла может быть организован по знакам «больше» и «больше или равно». Поэтому в некоторых случаях число N в памяти не отобразится.

4. Выявление подозреваемых лиц

Несложно предугадать следующий этап «следствия»: поиск фрагментов программы, содержащих характерные байты, найденные в предыдущем пункте. Лучше всего воспользоваться командами «H» и «Y» «отладчика». Найденные адреса памяти желательно переписать на бумагу. Попутно можно осуществлять визуальный «отсев» наименее вероятных адресов (однако при этом можно «зевнуть» и то место в программе, ради которого устроен поиск). Чем длиннее последовательность искомых байтов, тем меньше количество «попутных» фрагментов, не несущих нужной информации, но содержащих такие же комбинации байтов. Кроме того, если вы выявите несколько независимых комбинаций, присутствующих в исходном фрагменте, то сравнивая карты распределения их по памяти, можно указать расположение фрагмента с большей вероятностью.

5. Следственный эксперимент

Теперь нужно выяснить, какой именно из найденных фрагментов входит в искомый блок. Конечно, можно внимательно изучить каждый из них, но лучше поступить иначе. Надо поочередно менять содержимое найденных байтов таким образом, чтобы при работе программы можно было понять, когда она проходит измененный участок. Например, из обращения к ячейке $C000 легко можно сделать обращение к $C030, и теперь машина будет издавать звуковой сигнал при проходе через соответствующее место программы; команду ISR можно перенаправить на подпрограмму BELL и т. д. Однако если вы заменили информацию не в том участке программы, то это приведет к непредсказуемым последствиям, вплоть до зависания машины. Поэтому перед проведением подобных экспериментов не поленитесь вынуть дискету из дисковода (или хотя бы откройте крышку НГМД) во избежание порчи информации на диске.

6. Досье на преступника

Найдя методом проб и ошибок изменяемый блок, не помешает проверить, насколько он соответствует той схеме, которую вы для него предложили. Необходимо выяснить назначение команд хотя бы в непосредственной близости от найденной последовательности байтов. Особенно это важно при косвенном поиске фрагмента.

Если назначение команд нельзя понять с первого взгляда (а обычно это именно так), рекомендую распечатать на принтере по крайней мере один экран дизассемблированных команд (или не полениться переписать его от руки, если

принтера нет). В бумажном листинге можно поставить комментарии, облегчающие понимание текста программы.

1. Напротив обращений к адресам, назначения которых вам известны (ячейки ввода-вывода, п/п СМ и ДОС, ячейки нулевой страницы) желательно указать, к какому устройству происходит обращение и его цель (опрос клавиатуры, включение фазы F10 НГМД, считывание позиции верха текстового окна, обращение к п/п печати символа и т. д.).

2. Все ветвления программы (особенно близкие условные переходы) желательно указать на листинге стрелками (от команды ветвления до команды с адресом, на который производится переход).

Программисты считают, что легче написать свою программу, чем разобраться в чужой. Поэтому данный этап является одним из самых сложных во всей цепи «расследования».

7. Разработка плана захвата

Теперь, когда вы собрали всю необходимую информацию о «преступнике», его пора «брать». Для начала ответьте на вопрос: будете ли вы в будущем пользоваться неизмененной программой? (Например, корректное введение нового режима в любой редактор не должно помешать его работе — в крайнем случае новшеством можно не пользоваться. И соответственно, не придется использовать неизмененный редактор. Если же речь идет о внесении таких изменений в программы общего назначения, которые в некоторых ситуациях не только не полезны, но и вредны, на вопрос надо ответить утвердительно.) Если вы ответили «нет», то будьте уверены: новый блок можно «вживить» в программу. Если «да», то перед вами два пути: или храните две версии программы (измененную и неизмененную), или создайте собственный драйвер, который при запуске будет сам вносить изменения в предварительно загруженную основную программу (например, для внесения небольших изменений в крупные программы, типа Бейсика, СМ или ДОС).

А. Новиков,
студент III курса МПИ

Окончание следует

М. СТЕПАНОВ

Использование экранных страниц для активной работы с большими объемами информации

На ПЭВМ «Агат» оперативная память разбита на области, называемые экранными страницами, каждая из которых может отображаться на экран в виде символьной или графической информации. Наличие экранных страниц дает возможность сразу заполнить их информацией, а затем переключать с помощью простого программного блока. Опишем несколько случаев, в которых переключение страниц является достаточно эффективным.

1. Распечатайте таблицу значений функции с большим количеством строк и организуйте просмотр этой таблицы по желанию пользователя.

Решение. Будем печатать значения функции через строку для лучшего восприятия информации. Как только очередная страница заполнится числовой информацией, производится переход на следующую страницу, которая предварительно очищается. После того как таблица напечатана, начинает действовать та часть программы, которая представляет собой блок управления отображением страниц на экран. Нажатие на клавишу «стрелка вниз» вызывает переход к следующей странице, на клавишу «стрелка вверх» — к предыдущей, <ВК> — окончание программы. Страницы замкнуты в кольцо, т. е. за последней страницей следует первая, а первой предшествует последняя.

```
10 REM ПЕЧАТЬ ТАБЛИЦЫ
20 K=1:NS=12:TEXT=12:HOME
30 FOR X=0 TO 6.28 STEP 6.28/60
40 HTAB 0:VTAB K
50 RIBBON=1:PRINT X;:HTAB 16
60 RIBBON=6:PRINT SIN(X)
70 K=K+2:IF K>29 AND NS<15 THEN
                          GOSUB 1000
80 NEXT X
90 REM УПРАВЛЕНИЕ СТРАНИЦАМИ
100 GOSUB 2000
110 TEXT=0:HOME
120 END
1000 REM ПЕРЕХОД НА НОВУЮ СТРАНИЦУ
1010 K=1:NS=NS+1
1020 TEXT=NS
1030 HOME
1040 RETURN
2000 REM БЛОК УПРАВЛЕНИЯ
2010 NB=12
2020 TEXT=NS
2030 GET S$:KD=ASC(S$)
2040 IF KD=26 THEN NS=NS+1:
                  IF NS>15 THEN NS=12
     2050 IF KD=25 THEN NS=NS-1:
                  IF NS>12 THEN NS=15
     2060 IF KD=13 THEN RETURN
2070 GOTO 2020
```

2. Организуйте меню с большим числом предлагаемых вариантов. Перемещение указателя по меню осуществляйте клавишами-стрелками, а выбор варианта — нажатием на <ВК>.

Решение. Расположите текст меню на нескольких экранных страницах. Перемещение указателя после нажатия на вертикальные стрелки должно приводить либо к переходу на новый пункт меню, либо к переходу на новую страницу.

3. Расположите связный текст на нескольких текстовых страницах, перемежая его иллюстрациями, расположенными на графических страницах.

Решение. Для переключения графических страниц следует использовать оператор POKE, что позволяет избежать очистки этих страниц.

```
10 REM ИНФОРМАЦИЯ О ПОРЯДКЕ
                  СЛЕДОВАНИЯ СТРАНИЦ
20 DATA T,8
30 DATA T,9
40 DATA G,10
50 DATA T,11
60 DATA M,3
70 DIM TP$(4),NM(4)
```

```
80 FOR I=0 TO 4
90 READ TP$(I),NM(I)
100 NEXT I
110 REM ЗАПОЛНЕНИЕ СТРАНИЦ
120 FOR I=0 TO 4
130 IF TP$(I)="T" THEN GOSUB 1000
140 IF TP$(I)="G" THEN GOSUB 1500
150 IF TP$(I)="M" THEN GOSUB 2000
160 NEXT I
170 REM БЛОК УПРАВЛЕНИЯ ПЕРЕКЛЮЧЕНИЕМ
                                СТРАНИЦ
180 GOSUB 2500
190 TEXT=0:HOME:END
1000 TEXT=NM(I):HOME
1010 FOR K=1 TO 29 STEP 2
1020 VTAB K:PRINT "ОЧЕНЬ СВЯЗНЫЙ
                  ТЕКСТ НА СТР. ";NM(I)
1030 NEXT K
1040 RETURN
1500 GR=NM(I):COLOR=1
1510 FOR X=0 TO 62 STEP 2
1520 PLOT X,0 TO X,63
1530 NEXT X
1540 RETURN
2000 MGR=NM(I):COLOR=5
2010 FOR X=0 TO 126 STEP 2
2020 PLOT X,0 TO X,127
2030 NEXT X
2040 RETURN
2500 I=0
2510 IF TP$(I)="T" THEN TEXT=NM(I)
     2520 IF TP$(I)="G" THEN
                POKE $C700+NM(I)*4,0
     2530 IF TP$(I)="M" THEN
                POKE $C701+NM(I)*16,0
2540 GET S$:KD=ASC(S$)
2550 IF KD=13 THEN RETURN
2560 IF KD=25 THEN I=I-1:GOTO 2590
2570 IF KD=26 THEN I=I+1:GOTO 2590
2580 GOTO 2540
2590 IF I<0 THEN I=4
2600 IF I>4 THEN I=0
2610 GOTO 2510
```


А. НОВИКОВ

Еще раз об образах

В журнале ИНФО № 3—4 за 1992 г. был опубликован редактор для создания графических образов на ПЭВМ «Агат». Мной был разработан собственный редактор образов. В отличие от уже опубликованного, он рассчитан на более квалифицированного пользователя, позволяет упростить его работу и сократить затраты памяти и время вывода образа на экран.

Сеанс работы с редактором выглядит так.

При запуске, после завершения редактором подготовительных операций, он высвечивает заставку. Начало работы происходит после нажатия любой клавиши.

На экране появляются синие квадраты, символизирующие узловые точки будущего образа. В центре экрана мигает курсор. Редактор находится в режиме свободных перемещений, в котором запоминание вводимых команд не производится. Пользователь может нарисовать задуманный образ, опробовать методы его обхода. В этом режиме можно не бояться делать ошибки — они никак не отразятся на готовой продукции. Этот режим также может использоваться при создании образа для перехода от одной его части к другой, если какая-либо его часть не помещается на экране и изображена отдельно.

Управление курсором:

стрелки — перемещение со следом;

2,4,5,6 (боковые) — перемещение без следа;

<РЕД> — циклическая смена режимов СВОБ. ПЕРЕМЕЩЕНИЕ/ВВОД.

Для окончания работы с этим режимом курсор приводится в ту точку, с которой вы хотите начать (или продолжить) обрисовку образа, и нажать <РЕД>.

Основной режим — режим ВВОДА образа. В этом режиме все действия пользователя записываются в память в формате графического образа. Управление курсором осуществляется так же, как и в предыдущем режиме. В нижней строке экрана, помимо относительных координат курсора, выводятся поля байта: A, B и C. Именно наличие контроля этих полей позволяет оператору сэкономить память. Дело в том, что как поля, так и перемещения не совсем равнозначны между собой. На них распространяются 3 ограничения:

1) В поле C не может быть введено «перемещение со следом».

2) Байт должен быть «закрыт», то есть его последнее рабочее поле не должно содержать «перемещение вверх без следа».

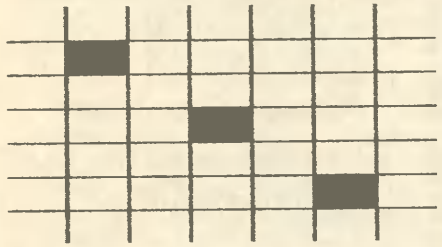
3) В случае невозможности (или ненужности) использования последних одного или двух полей (C или BC в силу п. 1 и 2), они должны быть сделаны нерабочими. Для этого достаточно в соответствующем поле нажать <ВВОД> (но не в поле A, так как это приведет к завершению ввода образа).

Эти ограничения продиктованы особенностями интерпретатора языка BASIC и рас-

сматриваются подробнее в ИНФО № 3—4 1992 и в описании ПЭВМ «Агат» Фг.00003—01 33 02. Редактор учитывает вышеизложенные ограничения и защищает работу пользователя от возможных ошибок. Приведу пример:

Пример 1. Пусть необходимо создать образ, изображенный на рис. 1, и курсор находится в нижнем квадрате. Если редактор не контролирует поля байта, то пользователь создаст образ и займет 6 байтов.

1



Если же редактор сообщает пользователю, какое поле сейчас вводится, то последний может изменить последовательность перемещений и уложиться в 5 байтов.

После ВВОДА в поле А образ считается законченным. Программа переходит к режиму просмотра. Теперь пользователь видит образ таким, каким он будет после «вживления» в ту программу, для которой он создавался. Стрелками вверх/вниз изменяется масштаб; стрелками вправо/влево — поворот образа; РЕД — переход к последнему режиму — режиму выдачи результата.

В этом режиме программа сообщает адреса памяти, в которые загружены коды образа для возможной записи их в виде B-файла, и выводит эти коды на экран в форме, удобной для макроассемблирования. В приведенном выше примере на экране появится сообщение:

ВАШ ОБРАЗ ЗАГРУЖЕН В ЯЧЕЙКИ $3000. $3004 И КОДИРУЕТСЯ ТАК:
!$CE0180400

Теперь можно загрузить программу, для которой создавался этот образ (или набрать NEW, если ее еще нет), и ввести в нее строки кодов образа, пронумеровав их. Это может выглядеть так:

```
10  BUFOBR = $1800: KOLVO = 1
20  * $E8:
            ! = BUFOBR
            ! :
30  * BUFOBR:
            ! = KOLVO
            ! $0400
90  REM ДАЛЕЕ САМ ОБРАЗ
100         ! $DCE0180400
990         ! :
1000  HGR= 2: COLOR= 15
1010  ROT= 0: SCALE= 1
1020  DRAW 1 AT 127,127
```

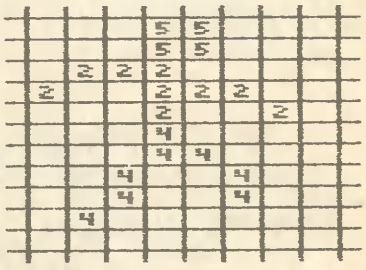
Здесь: BUFOBR — буфер образов — место в памяти, куда образы будут загружены; KOLVO — количество образов.

В строке 10 устанавливаются буфер образов и их количество; в 20 — интерпретатор настраивается на таблицу образов; в 30 — в таблицу заносятся количество образов и KOLVO раз смещение начала каждого образа относительно BUFOBR (разность между BUFOBR и адресом начала соответствующего образа). В строках 100—980 записываются коды образа (если их несколько, то образы записываются один за другим, начало каждого отмечается меткой для вычисления смещения — см. программу 2). Строки 1000—1020 отвечают за вывод первого образа в центре экрана.

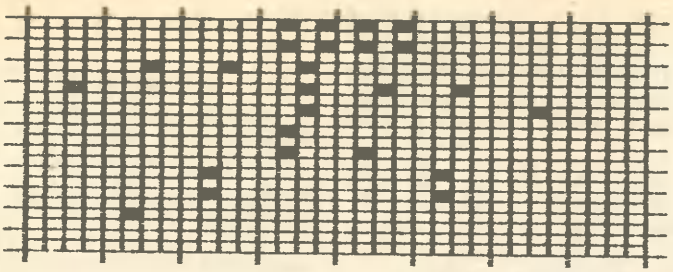
Эта программа является схемой наиболее простого и удобного оперирования образами.

Система управления образами работает только на графике высокого разрешения (HGP). Однако ее можно применять и для графики среднего разрешения (MGP), хотя это и уменьшает гибкость системы. При этом образ описывается в представлении HGP с учетом цвета будущего фона (машина не знает о том, что работает с цветной графикой и выводит якобы на HGP совокупность точек, которая при рассмотрении с MGP образует задуманный цветной образ). Приведу пример.

2



Пример 2. На MGP необходимо изобразить рис. 2 а. Преобразуем его для HGP (если хотите увидеть это преобразование на машине, нарисуйте цветное изображение в графическом редакторе (например, GREDIT), а затем нажмите <СБРУПР>, ×$C723, <ВВОД>). В результате получим рис. 2 б). Преобразовав при помощи редактора черно-белое изображение в последовательность кодов образа, встроим ее в программу. Инициализировав MGP, воспользуемся операторами DRAW или XDRAW. Учтите, что, для того чтобы изображение



оказалось в координатах (X, Y), его следует заказать в координатах (X×4+A, Y×2+B), где параметр A(0÷3) следует подобрать экспериментально (он зависит от положения начальной точки обхода образа и постоянен для данного образа). Параметр X= =(X AND 127), а B=(X AND 128)/128. Ниже привожу программу использования образов на MGP:

```
10  BUFOBR = $1800: KOLVO =
    3
20  * $E8:
      !  =  BUFOBR
      ! :
30  *  BUFOBR:
      !  =  KOLVO
      !  =  T1 -  BUFOBR
      !  =  T2 -  BUFOBR
      !  =  T3 -  BUFOBR
40 K = 1
90  REM ДАЛЕЕ САМ ОБРАЗ
100   !  T1:$0D0D0D161F1F1
    F56
110   ! $DFFB1BDF13B29292
120   ! $924A694820604821
130   ! $602068094D6989B2
140   ! $DADBDB134D8A1606
150   ! $00
160   !  T2:$4929690D161F1
    F1F
170   ! $56DFFB1BDF938D4A
180   ! $4D2160494A094DB1
190   ! $DB92DA121616C4C0
200   ! $C0E3D8B316DF9304
210   ! $00
220   !  T3:$0D0D0D161F1F1
    F8E
230   ! $16DFFB9B6A49494D
240   ! $6909DFDBDB12164D
250   ! $F99BFA1BDF934D49
260   ! $094D910700
990   ! :
1000  MGR= 2: COLOR= 9: PLOT
    0,73 TO 127,73: COLOR=
    15
1010  ROT= 0: SCALE= 1
1020  FOR B = 0 TO 1: FOR
    I = 1 TO 252 STEP 4
1030 K = K + 1: IF K = 4 THEN
    K = 1
1050  XDRAW K AT I,126 + B

1060  FOR T = 1 TO 100: NEXT

1070  XDRAW K AT I,126 + B

1090  NEXT I,B
```

В заключение отмечу, что поворот и масштабирование образа на MGP нежелательны, так как вызывают искажение цветов.

Для классов КУВТ-86 (любых типов):

1) Пакет АЛЬТЕРНАТИВА-2 (КУВТ). Назначение — обеспечение изучения информатики. Состав — 7 полностью записанных дисков. Включает: Новый сетевой монитор, обеспечивающий все виды пересылок, ТУРБО-ПРОЛОГ, новейшие версии текстовых, музыкальных, графических редакторов, новую систему управления базами данных, клавиатурные тренажеры, электронные таблицы, исполнители, пакет новых игровых программ, подробную техническую и методическую документацию.

2) Пакеты учебных программ РОБОТЛАНДИЯ и ЛОГО, предназначенные для изучения основ информатики с младшими школьниками. Являются полными аналогами пакетов, хорошо себя зарекомендовавших на классах ЯМАХА.

3) Система программирования ТУРБО-ПАСКАЛЬ, аналог известной на IBM/PC.

Для класса УКНЦ (любых типов):

Пакет АЛЬТЕРНАТИВА-2 (УКНЦ). Назначение — обеспечение изучения информатики в старших классах. Состав — 5 полностью записанных дисков. Включает: файловый монитор типа NORTON. Сетевой монитор, обеспечивающий все типы пересылок по сети; текстовые, музыкальные, графические редакторы; систему управления базами данных; систему ТУРБО-ПАСКАЛЬ с графикой, транслирующую в коды; ТУРБО-ПРОЛОГ, электронные таблицы с графическим выводом, пакет игровых программ, подробную техническую и методическую документацию.

Для классов КОРВЕТ (любых типов):

Пакет АЛЬТЕРНАТИВА-2 (КОРВЕТ). Назначение — обеспечение изучения информатики в старших классах. Состав — 5 полностью записанных дисков. Включает: сетевой монитор, обеспечивающий быструю пересылку по сети операционной системы CP/M и все виды работы с сетью; текстовый, музыкальный, графические редакторы; систему управления базами данных, систему ПАСКАЛЬ с графикой, транслирующую в коды; электронные таблицы, систему ТУРБО-ПРОЛОГ, пакет игровых программ, подробную техническую и методическую документацию.

МЫ ОБЪЯВЛЯЕМ АМНИСТИЮ всем НЕЗАКОННЫМ и НЕЗАРЕГИСТРИРОВАННЫМ пользователям наших пакетов и просим их всех обратиться к нам за получением новых версий. Мы ОБЪЯВЛЯЕМ о своей готовности передать на льготных основаниях наши инструментальные и программные средства учителям и методистам, способным создать с их помощью для нас новые программные и методические разработки.

Мы объявляем о своем выходе из предприятия АЛЬТЕРНАТИВА при НИИ АПН СССР.

Наша ставка — на высокое качество и большое число заказчиков, а не на высокие цены. Наши пакеты — это то, что Вам доступно и так необходимо сегодня!

Не теряйте драгоценного времени. *ВЫШЛИТЕ НАМ ЗАЯВКУ СЕГОДНЯ ЖЕ!*

Наш адрес: 656057, г. Барнаул, а/я 2513. НОВАЯ АЛЬТЕРНАТИВА.
Директор Г р и ц е н к о А. Н.

О. БЕЛОЗЕРОВ, В. ДОБРЯКОВ

Ну и что, что мала память!

Одним из эффективных приемов решения проблемы малого объема ОЗУ БК-0010 (в составе КУВТ-86) является оверлейное построение программ, так что в нужный момент в ОЗУ подгружаются блоки, необходимые в текущий момент времени для работы.

Основная часть ППС создается на Бейсике (Вильнюс-86), к числу очевидных недостатков которого следует отнести и невозможность использования операторов обращения к диску в программном режиме.

Вопросам памяти БК уделяется большое внимание. В частности, предложена следующая идея: обеспечить выполнение Бейсик-программы по заданному сценарию (Вычислительная техника и ее применение. 1989. № 3, 4; 1990. № 8). К сожалению, сценарий имеет жесткую структуру, не предусматривающую ее изменения в ходе выполнения.

Нам удалось устранить данный недостаток. Взяв за основу программу И. П. Березенцева и незначительно изменив способ управления, мы практически забыли о малой памяти БК, так как получили возможность создавать на Бейсике достаточно разветвленные блочные учебные программы, имеющие возможность из одного блока загружать другой, запускать его, задавать последовательность выполнения программных модулей.

На листинге управляющая подпрограмма начинается со строки 5000. Перед обращением к ней необходимо определить переменную A$, придав ей значение последовательности выполняемых команд с включением управляющих кодов, например код 10 — ВК (здесь и далее все значения кодов и адресов десятичные).

Учитывая возможные ошибки, возникающие в Бейсик-системе при работе с символьными переменными, лучше всего это сделать, определив соответствующую функцию пользователя (строки 90, 100).

В приводимом листинге из базовой программы вызывается модуль «TERMO 1», при этом символьная переменная A$ принимает следующее значение (последовательность команд):

```
10 REM ******************************
20 REM *                            *
30 REM * ОСНОВНАЯ ЧАСТЬ ПРОГРАММЫ   *
40 REM *                            *
50 REM ******************************
60 ......
70 ......
80 REM <<< ЗАДАНИЕ
          ПОСЛЕДОВАТЕЛЬНОСТИ КОМАНД >>>
90 DEF FNA$(A$)="NEW"+CHR$(10)+
     "CLOAD"+CHR$(34)+A$+CHR$(34)+",R"+
                          CHR$(10)+CHR$(3)
100 A$=FNA$("TT:TERMO1")
110 GOTO 5000
120 ......
130 ......
5000 REM <<< ПОДПРОГРАММА ЗАГРУЗКИ >>>
5010 DEF USR1=16128
5020 DEF USR2=16142
5030 DATA 5443,4930,5569,16222,
                        -27503,32450,135
5040 DATA 2551,2,289,5623,16222,66,
                          5599,16162,24
5050 DATA 135,9718,-18418,16,770,
                        95,-32694,5047
5060 DATA 40,-6665,2,34,9727,-30714,
                        28,757,-24640,24
5070 DATA 2743,20,9664,3,769,2,5599,
                      -32694,24,95,-24420,0
5080 I%=16128
5090 READ A%
5100 POKE I%,A%
5110 I%=I%+2%
5120 IF A% THEN 5090
5130 A$=USR1(A$) 'ПЕРЕСЫЛКА
                              ПЕРЕМЕННОЙ A$
5140 A%=USR2(A%) 'ЗАПУСК КОМАНД
                              НА ВЫПОЛНЕНИЕ
```

Если после выполнения вызываемого программного модуля необходим возврат в исходную программу, то символьную переменную A⍰ необходимо определить так:

```
NEW <ВК>
CLOAD "TERMO1",R <ВК>
```

Внимание! В конце значения переменной A⍰ обязательно должен стоять управляющий код 3!

Базовый программный модуль, их которого передается управление, должен сохранить свою работоспособность (или восстановить работоспособность) при выполнении команды NEW. Для этого кодовый блок необходимо располагать либо в буфере ввода-вывода с адреса 16128 (см. листинг), либо в младших адресах стековой области с адреса 256. Выбор места следует производить исходя из следующих соображений: если используется буфер ввода-вывода, то в вызываемом программном модуле должны отсутствовать операторы PRINT и INPUT#, нельзя также использовать команды LOAD и SAVE. Если же предпочтение отдано стековой области, необходимо учесть, что разрушение кодового блока может произойти при закрашивании сложных контуров оператором PAINT.

Чтобы перейти в стековую область, в приведенном листинге число 16222 (строки 5030 и 5040) нужно заменить на 350 и число 16162 (строка 5040) — на 290.

Кроме того, надо изменить следующие строки:

```
NEW <ВК>
CLOAD "TERMO1",R <ВК>
NEW <ВК>
CLOAD "<имя исходного файла>",R <ВК>

5010 DEF USR1=256
5020 DEF USR2=270
5080 I%=256
```

Если возникает необходимость в применении «жестких» сценариев, следует исключить строку 5140, задать символьную переменную A⍰ и после запуска подпрограммы записать сформированный новый управляющий блок на диск. При размещении в буфере ввода-вывода или в стековой области использовать соответственно команды

```
BSAVE "TT:<имя>",16142,16384
BSAVE "TT:<имя>",270,512
```


Более точно адрес конца программы подсчитывается для конкретного случая.

Запуск сценариев на выполнение производится командой

```
BLOAD "TT:<имя>",R
```

Е. ВОЛОДИН

БК-стихотворец

«Все мне пророчат счастье, но...
Вечная осень, багровые тона.
Устала я, устала я,
Хочу любить, все жду чего-то.
Вечная осень. Что ж — судьба!
Устала я. Надежда блекнет...»

Научить БК писать стихи совсем несложно, даже если он имеет только встроенный Фокал, не предусматривающий обработку символьных переменных.

Предлагаю программу, которая доставит немало удовольствия начинающим поэтам и графоманам. Она состоит из двух частей. Первая (строки 1.3—1.7) — создание словаря, вторая — «сочинение» и вывод на экран дисплея готовых стихотворений.

На запрос программы «СОЗДАНИЕ [1] или ИСПОЛНЕНИЕ [2]?:» выберите режим 1 и начинайте вводить слова, не забывая после каждого слова нажимать клавишу ⟨ВВОД⟩. Клавиша ⟨Q⟩ — конец работы.

```
1.10 X FCHR(12);A "СОЗДАНИЕ [1] ИЛИ
     ИСПОЛНЕНИЕ [2]?",A;X FCHR(12)
1.20 I (1-A)1.8
1.30 S I=0;S J=1;S W=0;S P=0;S L=1
1.40 S I=I+1;S A(I,J)=FCHR(FCHR(-1))
1.50 I (A(I,J)-81)1.60,2.60
1.60 I (A(I,J)-13)1.40,1.70,1.40
1.70 S N(J)=I-1;S J=J+1;S I=0;T " ";G 1.40
1.80 A "ФОРМАТ",X,"ДЛИНА",Y;X FCHR(12)
1.90 T "   ОПУС N ",%2.00,L,!!
2.10 S J=FABS(FITR(FRAN()*N))+1
2.20 F I=1,N(J);X FCHR(A(I,J))
2.30 S W=W+1;T " ";I (W-X)2.10;T !
2.40 S P=P+1;S W=0;I (P-Y)2.10;T !
2.50 S L=L+1;S P=0;I (FCHR(-1)-13)2.60,1.90
2.60 Q
```

Созданный массив сразу же запишите на магнитную ленту командой L O ИМЯ. Затем по команде M 2.1 войдите в режим редактирования строки и вместо переменной N подставьте число введенных слов. Снова запустите программу и переходите в режим 2. Запрос «ФОРМАТ»: обозначает требуемое количество слов в строке, а «ДЛИ-

НА» — общее количество строк. Ознакомившись с первым образцом «творчества» БК, нажмите клавишу ⟨ВВОД⟩, компьютер выдаст следующее стихотворение и т. д.

Для получения более или менее осмысленных стихов требуется тщательный отбор слов: рекомендуется вводить не отдельные слова, а смысловые блоки, внутри которых строго соблюдены правила грамматики. Лучше всего удаются «белые» стихи абстрактного философского содержания (см. эпиграф), но можно получить и рифмованные стихотворения. Для этого нужно заранее позаботиться о ритме (например, «наша Таня громко плачет», ударение везде на первый слог и организовать в программе проверку кодов окончаний слов.

И еще одно замечание: «поэтический талант» БК (впрочем как и у людей) существенно зависит от словарного запаса.

Примечания: переменная I — порядковый номер буквы в слове с номером J; M (J) — общее количество букв в J-м словаре; W — счетчик слов в строке; P — счетчик строк; L — номер стихотворения.

В. ВОРМСБЕХЕР, А. САЯПИН

Опыт рационального решения локальной сети с использованием IBM PC и БК-0011М

Анализ коллективного использования вычислительной техники в школах, институтах, малых (и не малых) предприятиях выявил ряд интересных особенностей.

Так, в подавляющем большинстве случаев (более 80 %) достаточно мощная вычислительная техника — персональные компьютеры типа IBM, Apple, Macintosh используются крайне нерационально и гораздо ниже заложенных в них возможностей. В первую очередь это связано с использованием такой вычислительной техники вне локальных сетей. Уже одно это резко снижает доступность техники для большого круга людей, заинтересованных в ней. Нередки ситуации, когда машина, способная одновременно обслуживать несколько десятков пользователей, занимается одним-единственным и целыми днями услаждает его играми.

С другой стороны, опыт, накопленный в школьных компьютерных классах по использованию локальных сетей отечественного производства, показывает, что основным их недостатком оказывается отсутствие достаточно мощных накопителей информации (жестких дисков) и невозможность использовать в их составе компьютеры, совместимые с IBM. Мы сознательно не касаемся остальных трудиостей, возникающих в повседневной эксплуатации школьных классов типа УКНЦ-01.01 и их предшественников — об этом уже исписаны впустую груды бумаги.

В данной статье нам хотелось бы поделиться опытом рационального использования вычислительной техники в рамках локальных сетей, не требующих больших денежных затрат, базирующихся на отечественных ЭВМ БК-0011М и работающих под контролем разработанной нами операционной системы, выгодно отличающейся от заводской своей надежностью и пользовательским комфортом.

Разрабатывая концепцию локальной сети, мы преследовали в первую очередь решение проблемы совместимости отечественной вычислительной техники, уже использующейся в локальных сетях и знакомой выпускникам наших школ, с достаточно мощной зарубежной техникой, имеющей емкие хранилища информации в виде винчестерских дисков. Мы также стремились обеспечить максимальную идентичность пользовательских интерфейсов всех наших программных средств с программами для IBM с целью минимизации проблем, связанных с переучиванием персонала. Помимо этого нам хотелось дать реальную и не очень дорогую возможность пользователям и той, и другой техники объединить доступность одних с оперативной мощностью других для достижения максимальной экономической эффективности имеющегося оборудования.

В качестве основы локальной сети используется штатная локальная сеть, применяемая в учебном компьютерном классе.

Отличительными особенностями разработанной локальной сети являются:

использование любого типа компьютера IBM PC (от XT до 486) в качестве центрального;

отсутствие каких бы то ни было переделок БК-0011М;

неиспользование нестандартного оборудования (за исключением контроллера параллельного интерфейса);

шинная структура сети, упрощающая прокладку кабеля;

скорость обмена по сети — 75 кбод;

связь скоростным параллельным интерфейсом центрального компьютера с одним из БК-0011М, используемым в качестве шлюза. Такое решение позволяет освободить центральный компьютер от управления сетью и обработок ошибок с одновременным уменьшением числа обращения к жесткому диску за счет буферизирования данных в ОЗУ шлюза;

центральный компьютер (IBM PC) может использоваться в своем обычном режиме наряду с одновременным обслуживанием запросов сети: программа обслуживания сети сделана резидентной и занимает в его памяти всего несколько килобайт;

переносимость файлов данных и текстовых файлов, введенных как на IBM, так и на БК-0011М, обеспечивается специальной программой — перекодировщиком READDWK;

число рабочих мест может наращиваться до 30 без подключения дополнительных плат к IBM PC;

удаление рабочих мест может достигать 300 м, что позволяет создавать распределенные управляюще-вычислительные системы;

центральный компьютер (сервер) может одиовременно функционировать в сети типа Novell NetWare.

Локальная сеть работает под управлением системы SPRUT, широко применяемой в обычных компьютерных классах на базе БК-0011М. Головная часть системы SPRUT исполняется на шлюзовом БК-0011 под управлением ОС БК-11. Периферийные модули системы представляют собой двухпанельное меню, аналогичное Norton Commander, и позволяют запускать на периферийных компьютерах не только программы в машинных кодах, ФОКАЛЕ или БЕЙСИКЕ, написанные для БК-0011М и БК-0010(-01), но и более сложные системы, разработанные для других, более мощных компьютеров. Эмулятор операционной системы поддерживает практически полностью языки высокого уровня: ПАСКАЛЬ, СИ, ФОРТРАН и т. д.

Распространение компьютеров фирмы IBM побудило коллектив разработчиков предприятия «ИНТЕРСЕРВЕР» создать целый набор средств, позволяющих преодолеть барьер несовместимости этих компьютеров с компьютерами серии БК в подавляющем большинстве пользовательских применений.

Так, программы перенормирования дискет позволяют переписать файлы с диска IBM на дискету для БК-0011М (или ДВК) как с перекодировкой, так и без нее. Это открывает возможность попеременной работы с текстовыми файлами на IBM и на БК-0011М. Такой же алгоритм работы возможен теперь и для файлов баз данных типа dBF. В ближайшем будущем станет возможным и перенос с БК на IBM исходных текстов, написанных на языке высокого уровня, с последующей их интерпретацией (или трансляцией).

Конечной целью коллектива является создание недорогого и вместе с тем высокоэффективного комплекта средств вычислительной техники с надежной центральной системой, имеющей большие вычислительные возможности, и развитой периферией интеллектуальных терминалов (до 30!).

Е. КУЗНИЦКИЙ, К. КАЗАРНОВСКИЙ, С. КРИЧЕВСКИЙ
Средняя школа № 491, Москва

Сетевая операционная система для учебных классов КУВТ-86М и УКНЦ-01

Любой пользователь современной ПЭВМ от системного программиста до издателя или бухгалтера имеет дело, прежде всего, с операционной системой (ОС). В то же время, как заводская поставка, так и распространяемые независимыми разработчиками пакеты программ для локальной сети, построенной на базе ПЭВМ БК-0011 и БК-0011М мало способствуют пониманию принципов функционирования ОС и навыков работы с операционными системами. Как уже справедливо отмечалось, в статье «Тема «Операционные системы» на базе КУВТ-86» данная тема имеет самостоятельный учебный интерес, но, кроме того, наличие ОС на каждой рабочей станции локальной сети значительно упрощает создание и эксплуатацию педагогических программных средств. В самом деле, до сегодняшнего дня разработчики были вынуждены писать свои программы на ассемблере или создавать на нем же библиотеки для имеющихся трансляторов с языков высокого уровня, поскольку существующие библиотеки используют функции и прерывания ОС, а при попытке сохранить результаты работы на дискетах, еще и разбиваться, в ассемблерной же и не отличающейся полнотой заводской технической документации (распространяемый НЦПСО сетевой монитор «Transit» облегчает последнюю задачу, но не обладает достаточной гибкостью). Такая работа больше подходит для системного программиста, а не для учителя или, тем более, ученика (возможность создавать серьезные программные продукты на Бейсике БК-0011 не рассматривается, как нереалистичная).

Технические возможности компьютеров БК-0011/11М (объем оперативной памяти, но, увы, не скорость передачи по локальной сети) позволили разработчикам описываемой системы пойти по пути, отличному от предложенного авторами упомянутой выше статьи и создать сетевую ОС, полностью поддерживающую работу ОС В11SJ на периферийных станциях.

Работа осуществляется следующим образом: на центральной машине запускается программа NEHOST.SAV, превращающая последнюю в файловый сервер, после чего включение периферийной машины для БК-0011М или инициализация локальной сети для БК-0011 приводит к загрузке в нее ОС В11SJ, использующей в качестве системного оригинальный сетевой драйвер NE.SYS таким образом, что системным устройством SY для этой машины становится NEO:. NEO сделан доступным только для чтения с тем, чтобы пользователи локальной сети не могли испортить или стереть программы общего пользования. Они, кроме того, могут работать с устройствами NE1:—NE7: и VD:. VD — это виртуальный диск, организованный с помощью оригинального драйвера VD.SYS и использующий недоступные для ОС В11SJ страницы оперативной памяти периферийной станции в качестве устройства с произвольным доступом для хранения информации, как на обычном диске. Он, разумеется, открыт для чтения/записи. Также открытые для чтения/записи устройства NE1:—NE7: представляют собой файлы на дисках BY0: и BY1: файлового сервера и работа с ними аналогична работе с логическими дисками LDn:, за исключением того, что присоединение файла в качестве логического диска осуществляется не командой MOUNT операционной системой, а программой NEDISK.SAV (запускаемой с периферийной машины), которую, впрочем, можно вызывать как команду ОС. Поскольку каждый файл может быть присоединен в качестве логического устройства только к одной станции, пользователи не имеют возможности, преднамеренно или нет, портить данные друг друга.

Отметим еще ряд дополнительных возможностей и особенной сетевой ОС.

Утилита NESTR.SAV автоматически выбирает продолжение стартового файла (вторичный стартовый файл) в зависимости от устройства загрузки.

Структура сетевого логического диска полностью совместима со структурой обычного логического диска, полученного с помощью драйвера LD, так что оператор центральной машины, например, учитель, имеет доступ ко всем файлам на дискетах.

Программа NEHOST позволяет использовать в качестве системной для периферийной станции дискету, отличную от используемой, в качестве таковой для центральной, что дает возможность, в частности, объединять в одну сеть машины двух типов: БК-0011 и БК-0011М. Кроме того, отладоч-

ный режим, реализованный в программе NEHOST, печатает всю необходимую информацию о каждой сетевой операции.

Хотя логически содержимое NEO: совпадает с SY: центральной станции, начальные блоки последнего подменяются программой NEHOST так, что команды загрузки ОС выполняются корректно.

Во время работы сети вся свободная память файлового сервера отводится под буфера чтения/записи, что несколько ускоряет ее работу, поскольку физически запись на диск осуществляется, по возможности, во время отсутствия сетевых запросов, а чтение часто требующихся блоков не приводит к фактическому обращению к диску.

В следующей версии сетевой ОС планируется реализовать, во-первых, сжатие передаваемой по сети информации, чтобы сделать скорость работы локальной сети более приемлемой, а во-вторых, систему доступа к файлам — логическим дискам, предусматривающую пароли, права доступа и все, что в таких случаях полагается.

Примечание. Ко времени выхода настоящего номера журнала работа над новой версией сетевой ОС должна быть завершена.

С. КОНОНОВ

«ЛИнТех» представляет: сетевая операционная система «NET—CP/M» версия А-1.0

В 1991 г. лабораторией информационных технологий («ЛИнТех») разработана сетевая операционная система для КУВТ «Корвет», «Форманта», «Нейва», «Орбита». Цель этой публикации — ознакомить в общих чертах всех пользователей вышеназванных КУВТ с, несомненно, самой необходимой системной программой, позволяющей в корне изменить характер работы на рабочих местах учеников (РМУ).

Программные средства, поставляемые в комплекте с КУВТ, не позволяют обеспечить нормальную работу на РМУ в стандартной локальной сети с дисками рабочего места преподавателя (РМП). Реально тот максимум, который можно сделать с помощью поставляемых сетевых средств — это обмениваться программными файлами интерпретатора BASIC и запускать на РМУ несколько специально подготовленных программ.

В этом случае не совсем понятно, чему кроме программирования на языке BASIC, преподаватель может научить учеников. Очевидно, что отнюдь не каждый школьник собирается и может стать профессиональным программистом, а вот знание общих принципов работы компьютера, операционной системы, систем управления базами данных, электронных таблиц, текстовых редакторов, словом, всего того, что прочно входит в нашу жизнь и с чем после окончания школы бывший ученик может реально столкнуться на работе — объективно необходимо. Именно для обеспечения полноценной работы на РМУ в среде стандартной операционной системы CP/M-80 со стандартными программами, которые для ОС CP/M-80 весь мир за целое десятилетие придумал великое множество, и разработана сетевая операционная система NET-CP/M.

Итак, имея NET-CP/M ученик, работая на РМУ в операционной системе, полностью совместимой с CP/M-80, может пользоваться мощнейшими пакетами программ, такими как система управления базами данных dBASE, электронными таблицами SuperCalk и Multiplan, разнообразными текстовыми и графическими редакторами, запускать игровые программы, работать с системами программирования Си, Паскаль, Ада, PL/1, словом, делать все, что раньше можно было делать только на РМП.

Основными составными частями ОС NET-CP/M являются две большие управляющие программы РМУ и РМП (далее по тексту будет использоваться термин «монитор» — синоним термина «управляющая программа»).

Монитор РМП осуществляет загрузку монитора РМУ по сети с РМП, исполнение всех запросов РМУ на операции с дисками РМП, определение режимов доступа РМУ, замену дисков, обработку ошибок и т. д. Монитор РМП запускается и работает на РМП постоянно в процессе функционирования ОС NET-CP/M.

Монитор РМУ представляет собой операционную систему, полностью аналогичную ОС CP/M-80, функционирующую на РМУ, и способную посредством локальной сети общаться с монитором РМП и передавать ему запросы на операции с дисками РМП, которые монитор РМП и исполняет.

Теперь необходимо сказать несколько слов о доступных для пользователя РМУ накопителях информации (дисках). В комплект РМП входят два накопителя на гибких магнитных дисках (ГМД), каждый из кото-

рых имеет емкость 800К. Кроме того, как РМП, так и РМУ в принципе могут использовать так называемый электронный диск, представляющий собой область графического запоминающего устройства (ГЗУ). Операционная система (ОС) позволяет обращаться к электронному диску точно также, как и к накопителю на ГМД и, с точки зрения пользователя, электронный диск ничем не отличается от накопителя на ГМД за исключением очень высокой скорости доступа к данным на электронном диске, его несменяемости и разрушении данных на нем при выключении питания машины.

Схемотехнически «Корвет» допускает установку 192 Кбайт ГЗУ. Это позволило бы иметь электронный диск 144 Кбайт с сохранением всех графических возможностей машины или 192 Кбайт при подавлении графики. К сожалению, производители КУВТ решили иначе — в стандартном КУВТ машины имеют ГЗУ всего 48 Кбайт. Это не позволяет использовать электронный диск на РМП, т. к. при рестарте ОС именно эти 48 Кбайт ГЗУ всегда очищаются, но на РМУ ОС NET-CP/M позволяет эффективно использовать даже 48 Кбайт ГЗУ в качестве электронного диска.

Итак, на РМП есть три накопителя — два НГМД и, возможно, электронный диск. На РМУ есть только электронный диск.

При работе в среде ОС NET-CP/M каждый ученик на РМУ получает доступ к своему электронному диску и ко всем трем накопителям РМП. При этом имена устройств формируются следующим образом: собственный электронный диск РМУ называется диск <A>; накопители <A>, <B>, <E> РМП именуются теперь <B>, <C>, <F> соответственно.

Как уже говорилось выше, монитор РМП представляет собой программу, которая должна функционировать на РМП все время работы ОС NET-CP/M. Именно эта программа исполняет все запросы от РМУ на операции с дисками РМП, определяет среду доступа РМУ, обрабатывает ошибки, загружает монитор РМУ и т. д.

Первое, что необходимо сделать для работы сети — это запустить монитор РМП — командный файл с именем NETCPM.COM. Работа программы основана на применении средств многооконного интерфейса, что обеспечивает высокие сервисные возможности и наглядность отображения и ввода информации. Сразу после запуска программы NETCPM. COM на экране появляются три окна «Меню», «Trade Mark» и «Сообщения».

В окно «Trade Mark» выводится информация о фирме «ЛИнТех» — разработчике данной операционной системы. Окно «сообщения» предназначено для выдачи информации о состоянии системы, сообщений об ошибках и восстановлениях после них. В окне «меню» предлагается выбрать один из шести возможных режимов работы монитора РМП — «Обслуживание», «Загрузка», «IFM-доступ», «Рабочий диск», «Замена диска», «Конец работы».

Режим «Обслуживание» является основным режимом работы монитора РМП. Находясь в нем, монитор РМП последовательно опрашивает РМУ на наличие запросов к дискам РМП и при необходимости исполняет их.

Все остальные режимы работы РМП носят вспомогательный характер и обеспечивают ОС NET-CP/M высокий уровень сервисных возможностей по управлению работой учеников в сети.

Вполне естественно, что прежде чем на РМУ появится монитор РМУ, его надо туда загрузить с РМП. Этим целям служит режим «IFM-загрузка». Еще необходимо присутствие двух файлов на текущем диске — NETCPM. SYS и NETLDR. SYS, в которых и находится собственно монитор РМУ. Загрузка всех включенных РМУ будет производиться одновременно и занимает около 15 секунд.

Выше утверждалось, что кроме своего собственного электронного диска ⟨A⟩ пользователи РМУ имеют возможность обращаться к трем накопителям РМП, которые в этом случае называются ⟨B⟩, ⟨C⟩, ⟨F⟩. В общем, все верно, но...

Монитор РМП поддерживает систему разграничения доступа к накопителям РМП (естественно, со своим собственным накопителем ⟨A⟩ каждое РМУ может сделать все, что считает нужным). Именно эта система позволяет вести работу без конфликтов большому количеству пользователей.

Основные идеи системы разграничения доступа следующие:

любое РМУ может читать файлы с любого накопителя РМП;

для любого РМУ может быть запрещена или разрешена запись на каждый из трех накопителей РМП;

один из трех накопителей РМП имеет особый статус и называется «рабочий диск». При осуществлении чтения или записи на диск (если данному РМУ разрешено писать на него), операция производится только в назначенной для данного РМУ личной USER-области, доступ к кото-

рой имеет только данное РМУ. Принцип выделения USER-области аналогичен действию команды USER ОС CP/M-80;

все операции РМУ с нерабочими дисками РМП производятся в USER0;

любая РМУ может быть выведена из обслуживания, т. е. с точки зрения монитора РМП перестать существовать.

Смысл введения системы разграничения доступа очень прост. Рассмотрим, что происходило бы, если бы этой системы не было.

Предположим, что три ученика собираются произвести запись на диск ⟨C⟩ РМП своего файла, который у каждого из них находится на собственном электронном диске ⟨A⟩ РМУ. При этом все трое лихорадочно придумывают оригинальное имя файла. И все они придумывают очень оригинальное имя — «MYFTLE.TXT».

Далее, шустрее всех оказывается первый ученик — он записывает свой файл под именем «MYFILE.TXT» на диск ⟨C⟩; затем ту же операцию повторяет второй ученик, затерев, соответственно, файл первого — имена файлов-то одинаковые; ну а третий предварительно просмотрев диск ⟨C⟩ и обнаружив, что файл «MYFILE.TXT» уже существует, решил, что он его уже скопировал раньше, но просто забыл об этом. А в это время...

Этой и массы других, подобных, курьезных ситуаций не возникает в случае применения системы разграничения доступа.

Пусть РМУ первого ученика соответствует USER 5, РМУ второго — USER 3, а РМУ третьего — USER 1. Рабочим диском является диск ⟨C⟩.

Тогда первый ученик записал бы свой файл «MYFILE.TXT» в USER 5, второй в USER 3, а третий, просмотрев USER 1 диска ⟨C⟩, не обнаружил бы чужой файл «MUFILE.TXT и записал бы свой.

Так что система разграничения доступа крайне полезная штука.

Для управления разграничением доступа предназначены режимы «Рабочий диск» и «IFM-доступ».

Режим «Рабочий диск» позволяет только определить в качестве рабочего один из дисков РМП — ⟨B⟩, ⟨C⟩ или ⟨F⟩.

Существенно интереснее режим «IFM-доступ». Именно в этом режиме можно для каждого РМУ запретить запись на любой из дисков РМП, определить номер USER-области при обмене с рабочим диском для каждой из РМУ, ввести или вывести РМУ из обслуживания.

Изначально установлены следующие параметры: все РМУ могут писать только на диск ⟨C⟩, он же является рабочим диском. Номер USER-области рабочего диска, приписанный к каждому РМУ, совпадает с физическим номером РМУ.

Если вдруг появилась необходимость заменить диск в одном из накопителей, то не стоит просто вынимать старый диск и вставлять новый — как минимум, можно испортить информацию на дискете. Специально для этой цели существует режим «Замена диска».

Режим «Конец работы» комментариев не требует — завершение работы монитора РМП ОС NET-CP/M. В этом случае все РМУ бросаются на произвол судьбы и, до тех пор пока монитор РМП не будет загружен вновь, не смогут выполнять обращения к дискам РМП (правда, при этом они могут совершенно безболезненно работать автономно со своим собственным электронным диском).

ОС NET-CP/M приходится обрабатывать большое количество ошибок, возникающих в локальной сети передачи данных и в процессе обмена данными с накопителями РМП. При возникновении ошибки в окне «сообщения монитора РМП» появляется сообщение об ошибке и система полностью самовосстанавливается.

Теперь самое время перейти к описанию монитора РМУ, который, как уже говорилось, представляет собой операционную систему, функционирующую на РМУ, способную обращаться к накопителям РМП и полностью совместимую с ОС CP/M-80. Это означает, что любая программа, корректно работающая в среде стандартной CP/M-80, будет работать и на РМУ под NET-CP/M.

Карта памяти монитора РМУ полностью соответствует карте памяти стандартной CP/M-80, но при этом все программные модули монитора РМУ резидентны (в стандартной CP/M-80 — транзитны CCP и BDOS).

Система команд процессора команд консоли несколько расширена и изменена в лучшую сторону. Дополнительно введен редактор командной строки с системой памяти командных строк.

Вообще, с точки зрения пользователя, монитор РМУ — это «нормальная и изнутри и снаружи» операционная система CP/M-80, имеющая возможность обращаться к четырем накопителям: ⟨A⟩ — собственный электронный диск, ⟨B⟩ и ⟨C⟩ — накопители РМП и ⟨F⟩ — электронный диск РМП. При этом обращение к диску ⟨A⟩ носит свободный характер, а к накопителям РМП определяется системой разграничения доступа монитора РМП.

Процессор команд консоли монитора РМП имеет и может исполнять следующие резидентные команды:

FORMAT — форматирование электронного диска ⟨A⟩ РМУ;

SCREEN — определение режима использования 48 Кбайт ГЗУ;

DIR — просмотр оглавления диска;

ERA — удаление файлов;

REN — переименование файлов;

TYPE — просмотр текстовых файлов;

COPY — копирование файлов.

В отличие от стандартной версии CP/M-80 отсутствуют команды USER и SAVE и добавлены очень полезные команды FORMAT, SCREEN и COPY.

Естественно, можно запускать транзитные команды и переопределять текущий диск.

Рассмотрим каждую из команд подробнее.

Команда FORMAT предназначена для форматирования электронного диска ⟨A⟩ РМУ. Эту операцию необходимо делать один раз, сразу после загрузки монитора РМУ. При ее выполнении стираются все данные, находящиеся на электронном диске, поэтому пользоваться ею надо очень осторожно.

Если у РМУ электронный диск полного объема (144 Кбайт), то при перезагрузке монитора РМУ вся предыдущая информация на электронном диске ⟨A⟩ сохранится и выполнять команду FORMAT не следует.

Команда SCREEN позволяет определить режим использования 48 Кбайт графического запоминающего устройства и может быть использована в двух вариантах.

Вариант SCREEN N предписывает использовать 48 Кбайт ГЗУ по прямому назначению — для отображения графики. В этом случае эти 48 Кбайт использовать под электронный диск ⟨A⟩ не удастся. Если на РМУ применяется полная память ГЗУ 192 Кбайт, то тогда 48 Кбайт ГЗУ будет использоваться для графики, а оставшиеся 144 Кбайт для электронного диска.

Вариант SCREEN N предписывает использование 48 Кбайт ГЗУ под электронный диск и подавление функций графики. Кроме того, если на РМУ полный объем ГЗУ 192 Кбайт, то это означает использование всех 192 Кбайт в качестве электронного диска.

Команда DIR полностью соответствует команде DIR CP/M-80. Предназначена для просмотра оглавления дисков.

Примеры:

A⟩dir — просмотр текущего директория, директория диска ⟨A⟩;

A⟩dir c:*.txt — поиск в директории диска ⟨C⟩ всех txt-файлов.

Команда ERA полностью соответствует команде ERA CP/M-80. Предназначена для удаления файлов.

Примеры:

A⟩era*.txt — удаление с диска ⟨A⟩ всех файлов с расширением TXT;

A⟩era b:mufile.com — удаление файла mufile.com с диска ⟨B⟩.

REN — команда переименования файлов. Формат этой команды несколько отличается от стандартного формата CP/M-80 и соответствует более удобному формату MS-DOS.

Формат команды:

A⟩ren ⟨старое имя файла⟩ ⟨новое имя файла⟩

Имена файлов не должны быть групповыми (т. е. в полях имени и расширения не должно быть символов «*» и «?»).

Пример:

A⟩ren b:e.txt exam.txt — переименование файла e.txt на диске ⟨B⟩ в файле exam.txt.

Команда TYPE полностью соответствует команде TYPE CP/M-80 и предназначена для просмотра текстовых файлов. Имя файла не должно быть групповым.

Для приостановки выхода информации на экран используется последовательность CTRL-S. Вывод текста можно прервать по нажатию на клавишу STOP.

Пример:

A⟩type c:exam.txt — просмотр файла exam.txt на диске ⟨C⟩.

Команда COPY предназначена для копирования файлов между электронным диском ⟨A⟩ РМУ и дисками РМП. Команда COPY использует специальные протоколы сетевого обмена и позволяет проводить копирование файлов максимально быстро. По времени процедура копирования файлов с помощью команды COPY занимает в десятки раз меньше времени, чем аналогичное копирование с помощью команды PIP.COM ОС CP/M-80 или аналогичной. Именно для команды COPY конечная скорость передачи данных составляет приблизительно 1 кбайт/с.

Формат команды:

A⟩copy ⟨имя файла источника⟩ ⟨имя файла приемника⟩

Запрещается копирование файлов между дисками РМП, а также копирование с диска на тот же самый диск. Групповое имя может применяться только в имени файла источника при условии отсутствия имени файла приемника (в качестве имени файла приемника может использоваться в данном случае только имя диска).

Примеры:

A⟩copy exam.*b: — скопировать все файлы exam.* на диск ⟨B⟩;

A⟩copy exam.*b:dest.* — ОШИБКА;

A⟩copy c:m.txt e.txt — копирование фай-

ла с именем m.txt с диска ⟨C⟩ на диск ⟨A⟩ под именем e.txt.

Переопределение текущего диска полностью соответствует стандарту CP/M-80.

Примеры:

A⟩b: — теперь текущий диск ⟨B⟩;

B⟩c: — а теперь ⟨C⟩.

Запуск транзитной команды полностью соответствует стандарту CP/M-80. Имя файла не должно быть групповым и не должно иметь расширения.

Пример:

A⟩b:akva+ — запуск графического редактора «АКВАРЕЛЬ» с диска ⟨B⟩.

ОС NET-CP/M имеет средства прерывания выполнения прикладной программы на РМУ без перезагрузки монитора РМУ по сети. Для этой операции необходимо сбросить компьютер клавишей RESET совместно с нажатием на клавишу F3, после чего экран очистится и система перейдет в режим редактирования командной строки.

Восстановление системы произойдет только в том случае, если зависшая программа не испортила программный код монитора РМУ, иначе все-таки придется перезагрузить монитор РМУ с РМП.

Как уже упоминалось, процессор команд консоли содержит специализированный редактор командной строки. Это позволяет не только редактировать вводимую строку, но и вспоминать/запоминать ранее введенные командные строки.

Редактирование строки, а также работа с буфером запомненных строк осуществляется с помощью клавиш 4, 6, 1, 2, 8, DEL, INS, CLS дополнительного поля клавиатуры, клавиш BS, ESC, STOP, CR основного поля.

Процессор команд консоли монитора РМУ имеет еще одну приятную сервисную возможность — в одной командной строке можно задавать сразу несколько команд на выполнение и они будут выполняться по очереди.

В командной строке различные команды разделяются с помощью пары символов — пробела и «⟩».

Перед выполнением каждой команды из командной строки расширенного формата предварительно производится ее отображение на экране монитора (с дополнительным символом «+» в начале).

Примеры:

A⟩screen y⟩b:poker

A⟩dir a: ⟩dir b: ⟩dir c: ⟩dir f:

A⟩mtplus record #tbrb⟩linkmt record, paslib/s⟩record

Ну, а теперь хочется заострить внимание на некоторых характерных для ОС NET-CP/M чертах, выгодно отличающих ее от многих существующих аналогов.

За счет эффективного протокола сетевого уровня существенно повышена конечная скорость передачи данных в локальной сети. В версии A—1.0 она составляет приблизительно 1 кбайт/с. В сочетании с наличием в мониторе РМУ специальной резидентной команды COPY это позволяет осуществлять очень быстрые пересылки данных между дисками РМП и РМУ.

Все программное обеспечение РМУ в ОС NET-CP/M резидентно, поэтому не происходит нужных перекачек данных по сети по завершению каждой программы или по нажатию на C.

В ОС NET-CP/M избрана весьма эффективная стратегия обеспечения выполнения запросов РМУ на операции с дисками РМП. Это позволяет работать на РМУ полностью автономно и соответственно очень быстро до тех пор, пока не появится объективная необходимость обращения к РМП.

Значительно расширены сервисные возможности системы. Процессор команд консоли монитора РМУ содержит специализированный многофункциональный редактор командной строки, имеется буфер запомненных ранее введенных строк, применяется расширенный формат командной строки. Монитор РМП имеет мощные средства разграничения доступа РМУ к ресурсам РМП, использует многооконный интерфейс для ввода/вывода информации, позволяет заменять любой диск в процессе работы.

Монитор РМП работает в автоматическом режиме и не требует наличия оператора.

Применяется максимально надежная система обработки ошибок монитора РМП при работе с дисками и сетью. Все ошибки и сбои системы самоликвидируются.

Монитор РМУ обладает повышенной надежностью — при «зависании» программы на РМУ возможен рестарт системы без перезагрузки по сети.

Количество пользователей в системе ОС NET-CP/M увеличено до 15.

ОС NET-CP/M не требует никаких специальных форматов дисков. Работа производится в стандартном CP/M формате на стандартных CP/M дисках.

Приобрести ОС NET-CP/M и получить необходимые справки можно в фирме «ЛИнТех».

А. АХМАНОВ, А. НЕЧАЕВ, А. СКУРИХИН

Архитектура процессора KP580BM80A

Следующая важная группа — команды управления работой программы. Что это за команды? Любая программа реализует какой-то алгоритм. Часто используемым элементом алгоритма является проверка каких-либо условий и в зависимости от результатов проверки выполнение того или иного фрагмента программы. Команды управления работой программы несут в себе эту функцию.

Кроме того, редко, когда программа пишется как единое целое. С ростом ее объема становится все труднее и труднее в ней разобраться. К тому же в программе неизбежно появляются некоторые совокупности операций, которые необходимо выполнять в самых разнообразных местах программы. Обычно, такие совокупности операций называются подпрограммами, или функциями. Естественно, напрашивается решение разбить программу на мелкие части и обращаться к этим частям по мере надобности. Команды вызова подпрограмм и возврата из них также относятся к командам управления работой программы. Разберемся, что делают эти команды.

Мы помним, что при выполнении программы текущая команда считывается из ячеек памяти, адресуемых с помощью регистра PC — счетчика команд. После выполнения очередной команды содержимое этого регистра автоматически увеличивается на число байт, занимаемых кодом команды, и считывается следующая команда. При работе с программой, не разбитой на части, процессор постепенно перебирает все адреса один за другим.

Обращение к подпрограммам или другим частям программы требует нарушения размеренной жизни процессора и памяти. В этом случае говорят о передаче управления. При этом содержимое PC изменяется таким образом, чтобы соответствовать адресу первой команды подпрограммы.

Надо отметить, что команды управления работой программы можно разделить еще на два класса — команды условные и безусловные.

1. Команда безусловного перехода. Эта команда состоит из 3-х байтов. Первый — код команды, а два других — адрес того места, куда нужно передать управление. Мнемоника команды:

```
JMP  NN
```

Продолжение. Начало см.: ИНФО, 1992, № 1—3.

происходит она от англ. jump — прыгать. Как только процессор получает эту команду, он без лишних раздумий загружает число NN в регистр PC, и дальнейшее от него уже не зависит. Следующая команда будет считана из ячейки памяти с адресом NN.
Обратите внимание, что адрес того места, откуда совершается прыжок, нигде не записывается и, следовательно, сам процессор без нашей помощи его не найдет. Код команды имеет вид:

```
1 1 Ø Ø Ø Ø 1 1
N N N N N N N N
N N N N N N N N
```

Причем, второй байт команды соответствует младшему байту адреса, а третий — старшему.

Команда выполняется за 10 тактов, т. е. за 4 мкс. На флаги эта команда не действует. Мы научились путешествовать по программе, приказывая процессору прыгать в том или ином направлении по заданному адресу. Но этого слишком мало. Как найти выход из ситуации, когда надо переходить лишь при выполнении определенного условия? С помощью безусловных переходов тут ничего не сделаешь. Чтобы помочь нам в решении подобных проблем, в составе команд процессора есть команды переходов при выполнении условия.

Вопрос на сообразительность. Где в процессоре может храниться информация о выполнении условий? Сообразительные могут следующее предложение не читать. Для остальных скажем: в флаговом регистре. Итак, переходы по условию.

2. Команды переходов по условию. Спектр условий, имеющихся в нашем распоряжении, не слишком богат. В условных переходах используются 4 флага Z, S, P, C. Соответственно получается 8 условий:
NZ — не ноль (Z=0)
Z — ноль (Z=1)
P — положительное (S=0)
M — отрицательное (S=1)
PO — нечетное (P=0)
PE — четное (P=1)
NC — нет переноса (C=0)
C — перенос (C=1)
Мнемоника команды условного перехода образуется из объединения буквы J — от jump и сокращенного обозначения условия, приведенного выше. Так, например, команда перехода, если Z=1, имеет вид JZ NN, где NN, как и в случае безусловных пере-

ходов, суть адрес того места в программе, куда передается управление. Код этих команд занимает три байта:

```
1 1 c c c Ø 1 Ø
N N N N N N N N
N N N N N N N N
```

где 3, 4 и 5 биты первого байта кодируют тип условия:

```
c  c  c
Ø  Ø  Ø   NZ
Ø  Ø  1   Z
Ø  1  Ø   NC
Ø  1  1   C
1  Ø  Ø   PO
1  Ø  1   PE
1  1  Ø   P
1  1  1   M
```

При выполнении подобной команды процессор сначала анализирует, выполнено ли условие, и только после этого начинает манипуляции с регистром PC. Если условие выполнено, то в PC заносится число NN и таким образом происходит передача управления. В противном случае выполняется следующая по порядку команда программы.

Хотя эти команды пользуются флаговым регистром, но его содержания не изменяют. Рассмотрим пример. Пусть содержимое C флага равно 1. Тогда после выполнения команды

```
JC   1221H
```

в регистре PC окажется число 1221H и в следующем машинном цикле процессор начнет считывать команду, первый байт которой находится по адресу 1221H.

Выполняются эти команды за 10 тактов, т. е. за 4 мкс.

В принципе теперь набор имеющихся у нас команд достаточно полный — с их помощью можно реализовать любой алгоритм. Однако у рассмотренных выше команд есть существенный недостаток. Мы нигде не сохраняем адрес того места, откуда совершается переход. Это неудобно при работе с подпрограммами. Действительно, одна и та же подпрограмма, скажем, вывода символа на экран дисплея, может вызываться из самых различных точек программы и при этом после окончания работы подпрограммы управление должно возвращаться в исходную точку. Для организации подобных программ предусмотрены специальные команды.

3. Команда вызова подпрограммы. Мнемоника команды:

```
CALL   NN
```

и происходит от английского call — звать, операнд NN — соответствует адресу новой команды подпрограммы. При выполнении этой команды процессор загружает в регистр PC число NN и таким образом происходит передача управления. Но это не все. Дело в том, что при выполнении команды CALL текущее содержимое регистра PC записывается в область памяти стека (т. е. автоматически происходит операция PUSH PC). Другими словами, содержимое PC укладывается в верхушку стека: старший байт в ячейку памяти с адресом SP-1, младший байт в ячейку памяти SP-2. Здесь SP обозначает содержимое регистра SP-указателя стека. Само же это содержимое уменьшается на 2. Код команды занимает три байта.

```
1 1 Ø Ø 1 1 Ø 1
N N N N N N N N
N N N N N N N N
```

Выполняется команда за 17 тактов, т. е. за 6,8 мкс. Рассмотрим пример. Пусть в PC хранилось число 1234H, в SP было записано 4321H, а в ячейках памяти с адресами 1234H, 1235H и 1236H были записаны байты

```
CDH
Ø1H
23H
```

Тогда в начале выполнения команды процессор считает ее код из ячейки 1234H. Это код CDH. Процессор сразу поймет, что это код команды CALL и считает еще 2 байта — адрес подпрограммы. Таким образом, в регистре PC окажется число 2301H — адрес следующей команды в программе. Мнемоника команды:

```
CALL   2301H
```

Процессор поместит в PC новый адрес 2301H, а в верхушку стека старое содержимое PC. Это значит, что в ячейку памяти с адресом 4320H попадет байт 12H, а в ячейку памяти 431FH — байт 37H. В регистр SP будет записано число 431FH. (Напоминаем, если в действии команды что-то непонятно, надо ее изучить с помощью отладчика.)

Таким образом, процессор сохранил адрес того места в программе, куда нужно возвратить управление после выполнения подпрограммы. Для того чтобы это сделать, необходимо следить за состоянием регистра SP (особенно при пользовании командами PUSH и POP). Так, к моменту возврата из подпрограммы содержимое SP должно быть тем же, что и в начале ее выполнения. Несоблюдение этого требования может привести к «зависанию» машины.

Теперь рассмотрим процедуру возвращения из подпрограммы.

4. Команда возвращения из подпрограммы. Мнемоника команды происходит от англ. return, что означает возвращение, и имеет вид RET. Код команды:

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

При выполнении команды возврата процессор извлекает из верхушки стека адрес того места в программе, откуда вызывалась подпрограмма. Младший байт адреса считывается из ячейки памяти с адресом, равным содержимому SP, а старший байт из ячейки памяти с адресом SP+1. При этом содержимое SP увеличивается на 2. После этого считанный из верхушки стека адрес записывается в PC и тем самым происходит передача управления. Команда выполняется за 10 тактов, т. е. за 4 мкс. Если внимательно посмотреть на действие этой команды, видно, что ее можно было бы назвать POP PC.

Рассмотрим пример. Пусть в регистре PC был записан адрес 3456H. В SP хранится число 7000H. В ячейках памяти с адресами 7000H и 7001H — байты 18H и 01H соответственно. После выполнения команды RET в PC будет записано число 0118H в SP 7002H и управление будет передано по адресу 0118H, откуда и будет считана следующая инструкция. Наверное, вы уже начинаете понимать преимущества команд CALL и RET в организации подпрограмм. Нужно заметить, что это еще не все. В состав команд процессора входят также команды вызова подпрограмм и возврата из них по условию. Рассмотрим эти команды.

5. Команды вызова подпрограмм по условию. Эти команды выполняют передачу управления подпрограммам при выполнении указанного в коде команды условия. Условия имеют точно такой же вид, что и в случае команд jump.

Мнемоника рассматриваемых команд образуется из объединения буквы C (call) и мнемонического обозначения условия. Например, CZ NN — вызвать подпрограмму по адресу NN, если Z флаг равен 1. В целом, образование мнемоники аналогично случаю с условными переходами. Так и код команды занимает три байта:

```
1 1 c c c 1 0 0
N N N N N N N N
N N N N N N N N
```

где биты 3, 4, 5 первого байта кодируют тип условия. Это кодирование осуществляется аналогично случаю с условными переходами. Выполняется эта команда следующим образом. Сначала процессор анализирует условие. Если оно выполнено, то 2 байта операнда команды загружаются в регистр PC и таким образом передается управление, а его старое содержимое перемещается в верхушку стека: в ячейку памяти с адресом SP-1 — старший байт, в ячейку с адресом SP-2 — младший. Само содержимое SP уменьшается на 2. Если условие не выполняется, то считывается следующая по порядку команда в программе.

Выполняются команды: за 17 тактов, т. е. за 6.8 мкс, если условие выполнено, и за 10 тактов, т. е. за 4 мкс, если — не выполнено. Рассмотрим пример. Если в PC хранилось число 1000H, в регистре SP 5007H, C флаг равен 1, то при выполнении команды CC 1234H в PC окажется число 1234H, в ячейке памяти с адресом 5006H — байт 10H, в ячейке 5005H — байт 03H, в SP будет записано число 5005H. В результате управление будет передано подпрограмме, находящейся по адресу 1234H. На состояние флагов рассмотренные команды не влияют. Коль скоро мы вызвали подпрограмму по условию, попробуем вернуться из нее по условию же.

6. Команды возврата из подпрограммы по условию. Мнемоника этих команд образуется аналогично рассмотренным условным переходам и вызовам. Мнемоника условий та же, только меняется первая буква. Теперь это не J и не C, а R (return). Так мнемоника команды возврата при условии C флаг равен 1 имеет вид:

```
RC
```

Кодируются эти команды так же, как и выше рассмотренные:

```
1 1 c c c 0 0 0
```

биты 3, 4 и 5 задают по известным нам правилам код условия. Выполняются эти команды за 11 тактов, т. е. за 4.4 мкс, если условие выполняется, и за 5 тактов, т. е. за 2 мкс, если — не выполняется.

Рассмотрим пример. Если S флаг был равен 1, в регистре SP было число 6000H, в PC — 1111H, в ячейке памяти с адресом 6000H находится байт 18H, в ячейке с адресом 6001H — байт 25H, то после выполнения команды

```
RM
```

в PC будет загружено число 2518H, в SP число 6002H и управление таким образом будет передано по адресу 2518H.

Рассмотрим еще один тип команды вызова подпрограммы. Эти команды были придуманы для вызова подпрограмм обработки прерываний — их длина один байт. По этой причине их часто используют программисты для вызова наиболее часто

встречающихся подпрограмм, что уменьшает общую длину программы. Называют их командами рестарта.

7. Команды рестарта. Мнемоника команды имеет вид:

```
RST  p
```

Число p указывает, по какому адресу передается управление. P принимает значение от 0 до 7, что соответствует адресам 00H, 08H, 10H, 18H, 20H, 28H, 30H, 38H, которые кодируются с помощью битов 3, 4, 5 и кода команды. Код имеет вид

```
1 1 t t t 1 1 1
```

Биты 5, 4, 3 кодируют значение P следующим образом

```
а д р е с        P
0 0 0 0 H        0 0 0
0 0 0 8 H        0 0 1
0 0 1 0 H        0 1 0
0 0 1 8 H        0 1 1
0 0 2 0 H        1 0 0
0 0 2 8 H        1 0 1
0 0 3 0 H        1 1 0
0 0 3 8 H        1 1 1
```

При выполнении этой команды процессор записывает в верхушку стека содержимое PC старший байт в ячейку памяти с адресом SP-1 младший байт по адресу SP-2, а в PC загружает адрес в соответствии с номером рестарта. Содержимое SP уменьшается на 2.

Команда выполняется за 11 тактов, т. е. за 4.4 мкс.

Рассмотрим пример. Пусть в PC хранилось число 1234H, в SP число 8005H. Тогда после выполнения команды

```
RST  7
```

код которой 11111111B хранился по адресу 1234H, в PC окажется число 0038H, в SP — число 8003H в ячейке памяти с адресом 8004H число 12H, в ячейке с адресом 8003H — байт 35H. Почему не 34H, догадайтесь сами. Возвратиться из подобной процедуры можно по команде RET.

Рассмотрим теперь несколько иной способ передачи управления. Необходимость в нем возникает в случае наличия у программы нескольких входных точек, адрес которых заранее неизвестен. Например, это наблюдается в программе BIOS — управляющей работой железа, входящего в состав компьютера. У этой программы несколько входов. Один для работы при первом включении компьютера, второй при перезагрузке и т. д. При этом точно известен адрес только одного входа — остальные надо вычислять. Для организации подобной процедуры можно использовать команду RET, предварительно вычислив адрес входа в подпрограмму и записав его в стек. Однако наш процессор имеет более удобную для этой цели команду.

8. Комаида безусловного перехода по адресу, указанному в регистровой паре HL. Мнемоника этой команды отвечает скорее обозначению процесса загрузки содержимого HL в регистр PC

```
PCHL
```

При выполнении этой команды процессор переписывает содержимое HL в регистр PC. Таким образом, происходит передача управления. Код команды состоит из одного байта:

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

Выполняется команда за 4 такта, т. е. за 1.6 мкс.

Наконец, последняя группа команд — команды управления работой процессора. Эти команды называют по-разному. Например, специальными. Их лучше не применять, не понимая работы компьютера. В противном случае вы можете завесить его так, что придется производить холодный старт. Рассмотрим эти команды.

1. Команда разрешения прерывания. Мнемоника команды:

```
EI
```

от англ. Enable Interrupt (разрешение прерывания). Команда воздействует на триггер разрешения прерывания (процесс прерывания мы рассматривали выше, см. ИНФО № 1, 2), точнее, устанавливает его в состояние «1».

Код этой команды равен

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

Выполняется команда за 4 такта, т. е. за 1,6 мкс. На флаги эта команда не воздействует.

2. Команда запрещения прерывания. Мнемоника команды: DI

```
DI
```

от англ. Disable Interrupt (запрещение прерывания). Действие этой команды противоположно предыдущей — она сбрасывает триггер разрешения прерываний.

После выполнения этой команды процессор не будет обрабатывать прерывания. Конечно, заманчиво запретить их и избавиться от головной боли по поводу есть ли там нужные процедуры обработки прерываний или нет. Но прерываниями пользуются различные полезные устройства: клавиатура, дисководы, часы реального времени и т. д. Так что бездумное использование команды DI приведет к выводу этих устройств из работы.

Что будет дальше, не трудио сообразить. Код этой опасной, сугубо специальной команды:

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

Выполняется она за 4 такта, т. е. за 1,6 мкс и на флаговый регистр не воздействует.

3. Команда остановки процессора. Мнемоника этой команды происходит от англ. halt — остановить:

```
HLT
```

При выполнении ее процессор начинает выполнять бесконечный цикл из команды пустышки (рассматриваемой ниже). Вытряхнуть его из этого цикла можно либо при помощи холодного старта, либо с помощью прерывания, если оно предусмотрено и не запрещено. Код команды:

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

Выполняется она за 4 такта, т. е. за 1,6 мкс, после чего машина зависает. На флаги команда не воздействует.

4. Команда пустышка. Эта команда хотя и выполняется процессором, но при этом ничего не происходит, кроме течения времени. Поэтому эту команду часто используют для задержки работы процессора до какого-либо события. Мнемоника команды имеет вид:

```
NOP
```

что происходит от англ. No OPeration — нет операции. Код этой команды:

```
0 0 0 0 0 0 0 0
```

«из ничего и выйдет ничего». Эта команда отвлекает процессор от нормальной работы на 4 такта, т. е. 1,6 мкс. Поскольку эта команда вообще ни на что не влияет, то в этом смысле флаги не являются исключением. Рассмотрим пример. Пусть в аккумуляторе содержится байт 08H, в регистровой паре HL — 0010H, а в регистре SP — 1111H. После выполнения команды

```
NOP
```

в аккумуляторе будет по-прежнему 08H, в HL — 0010H, а в SP — 1111H. Вопрос на сообразительность, в каком же регистре изменилось содержимое? Ответ: в PC. На этой полезной команде мы закончим рассмотрение команд нашего процессора.

Продолжение следует

Е. ШЕЙКМАН

Обучающе-контролирующие программы для КУВТ «Нейва» («Корвет»)

Пакеты ОБУЧ1, Математика1, ОБУЧ3, Парус-Н, ПРОФ1 и Физика1 предназначены для систематического использования в учебном процессе. Они состоят из независимых программ, написанных на Бейсике. Программы имеют жесткую структуру и не подлежат изменению. Каждая программа содержит конкретный материал по конкретной теме и ориентирована либо на обучение, либо на опрос, либо на то и другое вместе. Но в каждом пакете эти задачи реализованы поразному с точки зрения построения урока, дружелюбности программы, легкости и простоты восприятия. Программы используются в составе КУВТ с передачей их по локальной сети.

Все пакеты, кроме Парус-Н, могут работать в МикроДос, CP/M и «Корнет». Все зависит от того, какой версией интерпретатора Бейсика вы пользуетесь. Имеют хождения различные версии одиого и того же интерпретатора (так как авторские права на программные продукты практически не защищены). Часто программы не работают с «чужой» версией или работают не так, как было задумано. С целью предотвращения подобных эксцессов рекомендуется использовать интерпретатор, включенный в каждый пакет. Эта проблема не встает лишь при работе с пакетом Парус-Н. Он предназначен для МикроДос и автоматически загружает нужный Бейсик.

В ОБУЧ1 входят в основном программы, контролирующие знания. С точки зрения программирования они очень просты; позволяют проводить занятия по математике, физике, химии. После загрузки любой из программ перед учеником поочередно появляются вопросы по определенной теме и варианты ответов к ним. Ученик должен выбрать верный ответ. В пакет также входят уроки Бейсика, включающие и обучение и контроль. Их можно использовать как на уроках информатики, так и при самостоятельном изучении языка.

Программы пакета Математика1 сделаны

на более высоком уровне и с точки зрения программирования, и с точки зрения пользовательского интерфейса. Они сгруппированы по темам в каталог, получить который на экране можно, загрузив головную программу. После выбора нужной темы соответствующая программа загрузится автоматически.

В пакете собраны программы, позволяющие исследовать графики различных математических функций и проводить контроль знаний учащихся. После выбора в каталоге одной из математических функций и режима работы (исследование или опрос) на экране появляется выбранная функция. Задав коэффициенты, учащийся видит, как чертится график (возможен одновременный вывод на экран нескольких графиков с различными коэффициентами). Затем учащийся либо исследует график, общаясь с компьютером, либо отвечает на поставленные вопросы. Закончив работу, учащийся снова видит на экране каталог и может сделать выбор.

В ОБУЧ3 собраны обучающие и опросные программы по нескольким темам английского языка, а также «Тренажер», с помощью которого ученик может совершенствовать свои знания, а преподаватель — проверять их. В программе идет работа со словарем. Преподаватель может осуществлять подготовку к уроку, изменяя и дополняя словарь.

В ПРОФ1 собраны программы по профориентации, оценивающие характер, психическое состояние, наклонности, быстроту реакции и т. д. Программы прорабатывались с привлечением специалиста по психологии. В начале каждой программы имеется ее краткое описание.

Парус-Н представляет собой набор программ по информатике, в игровой форме помогающих учащимся закрепить знания двоичной системы счисления, элементов логики, обучающих составлению простейших программ, дающих понятие о циклах, подпрограммах. В пакете имеется клавиатурный тренажер, обучающий слепому десятипальцевому методу работы, а также словарь языка Бейсик.

Обучающе-контролирующий пакет Физика1 предназначен для изучения некоторых тем электротехники, оптики, механики и ряда других разделов школьного курса физики. Все программы написаны на Бейсике в едином стиле и рассчитаны на самостоятельное изучение определенной темы с последующим контролем знаний. В обучающей части широко использована компьютерная графика. В пакет включено несколько демонстрационных программ по изучению физических законов и явлений, имеется программа поддержки лабораторной работы «Определение элементарных частиц по их трекам».

Для успешной работы с пакетами от учителя не требуется значительных компьютерных знаний.

Практика показала популярность пакета Парус-Н: в его программах присутствует элемент игры, и процесс изучения и закрепления довольно сложного материала становится увлекательным. Каждому начинающему заниматься информатикой работа с этим пакетом принесет большую пользу. Ведь многие считают, что, научившись писать команды Бейсика, они становятся программистами. Отнюдь! Для достижения этой цели необходимо в совершенстве овладеть логикой программирования, чему и способствует пакет Парус-Н. Поэтому, думаю, он всегда будет пользоваться популярностью у начинающих пользователей «Корвета».

У пакетов Математика1 и ОБУЧ1 есть общий недостаток: некоторые математические обозначения (квадратный корень и т. п.) не очень наглядны на экране (в наборе символов «Корвета» нет знака корня, и его приходится создавать из различных составляющих).

Недостаток контролирующих программ пакета ОБУЧ1 еще и в том, что вопросы и ответы каждый раз появляются в одном и том же порядке, так что, используя эти программы несколько раз, можно просто запомнить правильные номера ответов.

Названия пакетов даны по каталогу АО «Хорда» (см.: Информатика и образование. 1991. № 4).

Р. ХУДЯКОВ

Практика работы с периферийным процессором УКНЦ

Прежде чем программа будет работать в ОЗУ ПП, ее нужно туда загрузить. Рассказ об этой операции начнем с массива параметров (далее — МП), формат которого приведен на первой распечатке.

```
MASSIV: .Byte   0       ;В этом байте будет
                        содержаться код ошибки или 0
COMAND: .Byte   0       ;Команда (01-ВЫДЕЛИТЬ
            ПАМЯТЬ, 02-ОСВОБОДИТЬ ПАМЯТЬ, 10-ЧТЕНИЕ,
                                20-ЗАПИСЬ, 30-ПУСК)
TYPDEV: .Word   32      ;Устройство-периферийный
                                процессор
ADR$PP: .Word   0       ;Адрес в ОЗУ ПП
ADR$CP: .Word   0       ;Адрес в ОЗУ ЦП
LENGTH: .Word   0       ;Длина программы в словах
                                (не байтах !)
ADRMAS: .Word   MASSIV  ;Слово всегда содержит
                начальный адрес массива параметров
        .Word   401     ;Стоповый элемент
                (используется при передаче)
```

```
ZAGR:   .Enable LSB
        MOV     #ADRMAS,R0      ;Адрес МП в R0
        MOV     #5,R1
        BR      1$              ;Передать четыре байта
2$:     MOVB    (R0)+,@#176676  ;(поле ADRMAS и слово,
                                содержащее 401)
1$:     TSTB    @#176674
        BPL     .-4
        SOB     R1,2$
        RTS     PC              ;Вернуться
```

Команды «Выделить память», «Освободить память», «Чтение», «Запись», «Пуск». Для правильного выполнения команды «Выделить память» поле COMAND МП должно содержать код 01, TYPDEV — код 32, а LENGTH – длину загружаемой программы. Значения остальных полей игнорируются. После пересылки МП в поле ADR$PP ПП возвращает адрес, начиная с которого после команды «ЗАПИСЬ» и будет размещена пересылаемая программа. При повторном запросе выделение памяти производится из оставшейся свободной. Если размер программы превосходит размер свободной памяти в ПП, то в поле LENGTH возвращается длина свободной памяти, а поле MASSIV не будет равно нулю.

Для команды «Освободить память» поля ADR$PP и LENGTH должны содержать те же параметры, которые задавались при выделении данного сегмента, иначе команда не будет выполнена, а в поле MASSIV будет возвращено ненулевое значение.

Для команды «Чтение» («Запись») поле ADR$CP содержит адрес ОЗУ ЦП, откуда (куда) будет пересылаться программа. Для команды «Пуск» ADR$PP содержит стартовый адрес программы. Сама программа должна быть оформлена как подпрограмма (заканчиваться RTS PC) и быть перемещаемой, так как место ее размещения заранее неизвестно.

На распечатке 2 приведена подпрограмма загрузки МП в ПП.

```
ZAGPP:  .Enable LBB
        MOVB    #1,COMAND       ;Отработать команду
                                        "ВЫДЕЛИТЬ ПАМЯТЬ"
        MOV     (R5),OCP        ;Через R5 передается
                                        табличка из
        CALL    ZAGR            ;двух слов. 1-е - длина
    программы, 2-е - ее начальный адрес в ОЗУ ЦП
        TSTB    MASSIV          ;А памяти хватит ?
        BEQ     1$              ;Да
        MOV     #NOMEM,R0       ;Выведем сообщение об
                                        ошибке
        EMT     351             ;"Не хватает памяти"
        CMP     (R5)+,(R5)+     ;Увеличим R5 на четыре
        SEC                     ;Установим признак
                                        ошибки
        BR      2$              ;И на выход
1$:     MOVB    #20,COMAND      ;Отработать "ЗАПИСЬ"
        MOV     (R5)+,LENGTH    ;Из таблички 1-е слово
                                        в LENGTH,
        MOV     (R5)+,ADR$CP    ;2-е в ADR$CP
        CALL    ZAGR
        MOVB    #30,COMAND      ;Отработать "ПУСК"
        CALL    ZAGR
        MOVB    #2,COMAND       ;Отработать "ОСВОБОДИТЬ
                                        ПАМЯТЬ"
        MOV     -4(R5),ADR$CP
        CALL    ZAGR
2$:     RTS     R5              ;Вернуться
```

Она передает по каналу К2 адрес массива параметров и два байта завершения передачи, нормализующие состояние канала.

На распечатке 3 приведена полная под-

программа загрузки программы, ее пуска и последующего освобождения памяти ПП по окончании работы.

```
.MACRO  %ZAGPP  STADR,ENDADR
     JSR     R5,ZAGPP
     .WORD   <ENDADR-STADR/2>,STADR
.ENDM   %ZAGPP
```

Формат вызова подпрограммы приведен на распечатке 4.

```
     .TITLE  MYPROG
START:  %ZAGPP  STARTPP,ENDPP    ;Загрузить программу
     EMT     350              ;Выход в монитор
   ;Здесь находятся подпрограммы, сообщения и массив
                                      параметров
STARTPP: INC     (PC)+           ;Эту программу можно
                                   заменить любой другой
     .WORD   0                ;---X---X---X---X---
     RTS     PC               ;Вернуться в ЦП
ENDPP:
     .END    START
```

А вот полностью законченная программа.

```
┌─── АДРЕС
↓
!0000!0000!0000!0000! !0000!0000!0000!0000!
 зел.желт.фиол.красн.бирюз.зел.голуб.черн.
```

Итак, программа в ПП загружена.

Работа с регистрами внешних устройств не отличается от обычной. Адресное пространство ОЗУ ПП имеет диапазон 0—77777, ПЗУ ПП — 100000—177777, т. е. 32 Кбайт ОЗУ и 32 Кбайт ПЗУ. ПП (как и ЦП) имеет свой пультовый отладчик, облегчающий отладку программ (вход в него осуществляется из меню установок одновременным нажатием клавиш УПР и @). Работать с ним нужно так же, как и с пультовым отладчиком ЦП.

Основное внимание мы уделим дополнительным возможностям, которые могут быть использованы только при помощи ПП. Начнем со средств отображения. Для начала «пощелкаем» палитрой. В УКНЦ любой цвет, выводимый на экран, можно представить любым другим (поменять местами красный и зеленый, фиолетовый и бирюзовый и т. д.). Допустим, вы поменяли синий на белый. Теперь все, что вы запишите в голубой план, на экране появится белым цветом, причем если какая-либо другая программа не изменит значения соответствующей ячейки памяти, то даже после выхода в монитор все, что должно было бы отображаться на экране голубым, будет отображаться белым, включая графическую информацию. Замена цветов происходит очень быстро (практически мгновенно), на этом можно построить интересные эффекты.

Формат ячеек управления аппаратным отображением приведен на рисунке (таких пар слов четыре, их «полномочия» приведены в табл. 1). Каждый из цветов представлен четырьмя битами. Байты каждой из тетрад означают (слева направо):

яркость цвета (1 — 100 % яркости, 0 — 50 %);

красный цвет (1 — красный есть, 0 — нет);

зеленый цвет (1 — есть, 0 — нет);

голубой цвет (1 — есть, 0 — нет).

```
     MOV     #2470,R0         ;Адрес палитры экрана
                                        в R0
     MOV     (R0),-(SP)       ;Запомнить в стеке
                                   текущее значение
     MOV     2(R0),-(SP)      ;палитры
     CLR     (R0)             ;Выключить экран
     CLR     2(R0)            ;(все цвета на экране
                                   представить черным)
     MOV     #100000,@#177010
     MOV     #30.,R1
2$:  MOV     #20.,R2
1$:  MOVB    #377,@#177012    ;Нарисовать квадрат
     INC     @#177010
     SOB     R2,1$
     ADD     #<80.-20.>,@#177010 ;
     SOB     R1,2$
     MOV     #1,R1            ;"Щелкаем" палитру.
3$:  MOV     R1,R3
     .REPT   4                ;Четыре раза ASL R3
     ASL     R3
     .ENDR
     BIC     #360,(R0)
     BIS     R3,(R0)          ;Квадрат  отображается
     CLR     R2               ;цветами от голубого до
                                        белого
     SOB     R2,.             ;<-задержка
     SOB     R2,.             ;<-задержка
     INC     R1
     CMP     #8.,R1
     BNE     3$
     MOV     (SP)+,2(R0)      ;Возвращаем предыдущие
                                   значения
     MOV     (SP)+,(R0)       ;ячейкам
     RTS     PC               ;Возврат
```

На машинах литеры 5 завода «Квант» из схемы контроллера цветного монитора удалена ИС 155ЛП9, поэтому яркостный бит не оказывает влияния на изображение; на почти половине машин завода «СЭМЗ» ИС 155ЛП9 заменена на ИС 155ЛН2, поэтому в яркостный бит для получения 100 % яркости нужно записать 0, а для 50 % — 1. Все это справедливо только для цветного монитора, на отображение градаций яркости на черно-белом мониторе яркостный бит не влияет.

Отметим также, что в начале 1991 г. изменена схема подключения цветного монитора (поменялись местами выходы красного и зеленого цветов).

Таблица 1

Адрес	Зона, управляемая соответствующими ячейками
2400	Верхняя служебная строка
2470	Собственно сам экран
4700	Меню установок
6750	Нижняя служебная строка

Приведем программу для ПП, выполняющую следующее: экран выключается, на нем (невидимо) рисуется голубой квадрат; экран включается (квадрат появляется); затем цвет квадрата начинает меняться от голубого до белого.

```
        MOV     PC,R0
        ADD     #SOOB-.,R0      ;Вычислить адрес
                                        сообщения
        MOV     R0,AMT
        MOV     PC,R0
        ADD     #SOOB1-.,R0     ;То же
        MOV     R0,AMT1
        MOV     #135230,@#2400  ;Разрешить вывод на
                                        экран всех цветов
        MOV     #177334,@#2402  ;для верхней служебной
                                        строки
        MOV     #135230,@#6750  ;То же сделать и для
                                        нижней
        MOV     #177334,@#6752
        MOV     #1,@#22614      ;Цвет символов в
                                верхней строке - голубой
        MOV     #3,@#23106      ;Цвет символов в нижней
                                строке - бирюзовый
        MOV     #63146,@#22620  ;Цвет знакоместа в
                                верхней строке - желтый
        MOV     #52525,@#22622
        MOV     #52525,@#23112  ;Цвет знакоместа в
                                нижней строке - фиолетовый
        MOV     #42104,@#23114
        MOV     #67,@#2372      ;Верхняя строка в
                                        формате 4 (10X24)
        MOV     #47,@#6742      ;Нижняя строка в
                                        формате 3 (20X24)
        EMT     52              ;Вывести сообщение в
                                        верхнюю строку
AMT:    .WORD   @               ;Здесь находится адрес
                                        сообщения
        EMT     56              ;Вывести сообщение в
                                        нижнюю строку
AMT1:   .WORD   @               ;Здесь находится адрес
                                        второго сообщения
        RTS     PC              ;Возврат в ЦП
SOOB:   .ASCIZ  <40.>/"ТЕХНОКОМ"/
SOOB1:  .ASCIZ  <40.>"    ПРЕДСТАВЛЯЕТ    "
        .EVEN
```

При выводе информации в служебные строки можно менять формат и цвета символа и знакоместа, причем цвета меняются двумя способами. Первый — непосредственное изменение цветов символа и знакоместа, второй — через изменение палитры цветов вывода.

Пример программы с изменением цветов символа и знакоместа.

```
        MOV     PC,R0           ;Вычислить адрес
                                        сообщения
        ADD     #SOOB-.,R0
        MOV     R0,AMT
        MOV     PC,R0           ;То же
        ADD     #SOOB1-.,R0
        MOV     R0,AMT1
        EMT     52              ;Вывести "ТЕХНОКОМ" в
                                        верхнюю строку
AMT:    .WORD   @               ;Здесь адрес сообщения
        EMT     56              ;Вывести "ПРЕДСТАВЛЯЕТ"
                                        в нижнюю строку
AMT1:   .WORD   @               ;Здесь адрес второго
                                        сообщения
        MOV     #67,@#2372      ;Формат верхней
                                        строки - 4 (10X24)
        MOV     #47,@#6742      ;Формат нижней
                                        строки - 3 (20X24)
        MOV     #135236,@#2400  ;Для верхней строки
                                представляем желтый цвет
        MOV     #174734,@#2402  ;голубым, а черный
                                        желтым
        MOV     #135235,@#6750  ;Для нижней строки
                                представляем желтый цвет
        MOV     #175734,@#6752  ;бирюзовым, а черный
                                        фиолетовым
        RTS     PC              ;Возврат
SOOB:   .ASCIZ  <40.>/"ТЕХНОКОМ"/
SOOB1:  .ASCIZ  <40.>"    ПРЕДСТАВЛЯЕТ    "
        .EVEN
```

Ячейки с адресами 2400, 2402 (6750, 6752) отвечают за палитру вывода для верхней (нижней) служебной строки. Их формат был описан выше.

В ячейках 22614, 23106 хранится цвет символа в формате (0 — черный, 1 — голубой, 2 — зеленый, ..., 7 — белый), а в ячейках 23112, 23114 — цвет знакоместа в формате регистра кода цвета фона (177020, 177022) (этот формат будет рассмотрен ниже). В исходном состоянии (после включения питания) голубой план для вывода в служебные строки заблокирован.

Пример программы с изменением палитры вывода.

```
                4     3     2     1     - номер точки в октете
177020:  0000'0000'0000'0000
          RGB   RGB   RGB   RGB   - красный, зеленый,
                                        голубой планы

                8     7     6     5     Старший бит
177024:  0000'0000'0000'0000  каждой тетрады
          RGB   RGB   RGB   RGB   не используется
```

Требования при выводе текстовой информации в служебные строки таковы: первый байт любого сообщения есть байт позиционирования, а сам текст сообщения должен быть восьмибитным, так как коды ⟨РУС⟩ и ⟨ЛАТ⟩ (16(8) и 17(8) при печати не отрабатываются.

Теперь немного о выводе графической информации из ПП. Есть целая группа регистров, существенно облегчающая этот процесс:

177016 (запись/чтение) — регистр кода цвета точки. Формат — число от 0 до 7 (цвет точки);

177024 (запись) — регистр октета точки. Формат — восьмеричное число от 0 до 377 (восемь битов — восемь точек на экране; бит содержит 1 — точка зажжена, 0 — погашена);

177020 — регистр кода цвета фона (планы 0—1), 177022 — регистр кода цвета фона (планы 1—2). Оба регистра представляют собой единое целое; формат приведен на рисунке. Выведем на экран октет, в котором первая точка — черная, вторая — голубая, третья — зеленая, ..., восьмая — белая. Для этого нужно записать в 177020 число 0011 0010 0001 0000, а в 177022 — 0111 0110 0101 0100.

```
        MOV     #^B0011001000010000,@#177020 ;Загрузить
                                        регистр кода цвета
        MOV     #^B0111011001010100,@#177022 ;фона по
                принципу 1-я точка черная , восьмая - белая
        MOV     #100000,@#177010 ;Загрузить адрес
                                планов в регистр адреса планов.
        MOV     #<264.*80.>,R1  ;В R1 общее число байтов
                                        экранной зоны
10:     CLR     @#177024        ;"Чистим" текущий октет
        INC     @#177010        ;Увеличим на 1 адрес в
                                        планах
        SOB     R1,10           ;Все это в цикле
        ;НА ЭКРАНЕ ВИДНЫ ЦВЕТНЫЕ ПОЛОСКИ
        MOV     #^B0100010101100111,@#177020 ;Загрузить
                                        другие значения
        MOV     #^B0000000100100011,@#177022 ;цвета фона
        MOV     PC,R2
        ADD     #SPRITE-.,R2    ;Адрес спрайта в R2
        MOV     #100000,@#177010 ;Адрес в планах =
                                                100000
        MOV     #7,@#177016     ;Выводить будем белым
        MOV     #11,R0          ;Размеры спрайта
                                        (Y = 11, X = 21)
30:     MOV     #21,R1
        ;ЭТУ ЧАСТЬ ВЫДЕЛИМ
```

```
20:   MOVB    (R2)+,@#177024  ;Байты, из которых
                              состоит спрайт, по порядку на экран
      ;КОНЕЦ ВЫДЕЛЕННОЙ ЧАСТИ
      INC     @#177010        ;Увеличим адрес в планах
                                                        на 1
      SOB     R1,20           ;Следующий байт на экран
      ADD     #<80.-21>,@#177010 ;Отступить для вывода
                                    следующей строки
      SOB     R0,30           ;спрайта.
      RTS     PC              ;Возврат
      ;НИЖЕ ДАН САМ СПРАЙТ
SPRITE: .Byte 377,7,377,37,36,36,74,170,340,77,360,
                                    340,201,377,300,3,17
        .Byte 61,6,6,30,30,6,30,60,60,140,140,140,
                                    300,200,201,3,7
        .Byte 60,0,6,0,60,3,30,60,30,300,140,60,140,
                                    0,203,207,7
        .Byte 60,0,6,0,340,1,30,60,30,300,140,30,
                                    140,0,203,315,6
        .Byte 60,0,376,1,300,0,370,77,30,300,340,17,
                                    140,0,203,171,6
        .Byte 60,0,6,0,340,1,30,60,30,300,140,30,
                                    140,0,203,61,6
        .Byte 60,0,6,0,60,3,30,60,30,300,140,60,140,
                                    0,203,1,6
        .Byte 60,0,6,30,30,6,30,60,60,140,140,140,
                                    300,200,201,1,6
        .Byte 170,0,377,37,36,36,74,170,340,77,360,
                                    340,201,377,300,3,217
        .Even
ENDPRG:                         ;Конец программы в ПП
```

177026 — регистр маски планов. Формат — число от 0 до 7 (один или несколько планов, в которые нужно запретить запись информации). Регистр не блокирует запись в зеленый и красный планы со стороны ЦП.

Рассмотрим использование этих регистров. Приведенная ниже программа сначала заполняет экран цветными вертикальными полосами толщиной в одну точку, затем выводит в левом верхнем углу экрана спрайт.

```
20:   MOVB    (R2)+,R3
      MOV     R3,@#177024
```

Вывод информации на экран может быть двух видов — с прорисовкой новой информации по старому фону и со стиранием старой информации. Чтобы рисовать по старому фону, нужно к регистру 177024 адресовать байтно (MOVB), для стирания — словно (MOV). В приведенном выше примере прорисовка со стиранием обеспечивается заменой выделенной части программы на следующую:

```
      BIS     #6,@#177710     ;Тактовая чатота таймера
                                                  16 мкс
      BIC     #17400,@#177716 ;Очистить разряды с 8 по
                                                  12
      MOV     #200,R0         ;Установить в R0
                              единственный разряд - 7
10:   XOR     R0,@#177716 ;Если он есть в 177716 ,то
                                  снять его и наоборот
      MOV   #<от этого числа зависит высота звука>,R1
      SOB     R1,.            ;Задержка именно на этом
                                                  числе
      BR      10              ;Бесконечный цикл
```

Как видно, с помощью этой группы регистров можно существенно повысить скорость вывода графики, однако иногда и этой скорости не хватает. Можно еще в несколько раз ее увеличить, используя нерегистровый (прямой) доступ к видеоОЗУ. Но об этом в следующий раз.

Теперь немного о музыке. Звук из динамика УКНЦ можно получить при помощи регистров 177710 и 177716. В регистре 177716 разряды с 7 по 12 позволяют получить звуки с частотой от 60 Гц до 64 кГц при частоте программируемого таймера 2, 4, 8 и 16 мкс. В табл. 2 даны соответствия комбинаций разрядов 8—16 регистра 177716, а в табл. 3 — разрядов 1, 2 регистра 177710.

Частоты звука приведены для тактовой частоты таймера 16 мкс. Если же тактовая частота равна 8 мкс, то значения частот в правой колонке увеличатся в 2 раза (соответственно для 4 мкс в 4 раза, а для 2 мкс в 8 раз).

Если разряды 8—12 содержат 0, то частота звука будет зависеть от того, с какой скоростью разряд 7 переключается из 0 в 1. Способ извлечения звука прост. Меняя значение, загружаемое для задержки, мы управляем высотой звука. Программа, приведен-

Таблица 2

Разряды регистра 177716						Частота звукового сигнала
7	12	11	10	9	8	
Программно формируется	0	0	0	0	0	Соответствует программе
1	0	0	0	0	1	60 Гц
	0	0	0	1	0	250 Гц
	0	0	1	0	0	500 Гц
	0	1	0	0	0	1 кГц
	1	0	0	0	0	8 кГц
0	Звук отключен					
1	Комбинация логических значений					Сборка сетки частот по «И»
Программно формируется	*	*	*	*	*	Сборка сетки частот по «И» промодулирована программой

ная ниже, работает именно по этому принципу.

```
1-я октава
      До - 130,8
      До-диез, ре-бемоль - 138,6
      Ре - 146,8
      Ре-диез, ми-бемоль - 155,6
      Ми - 164,8
      Фа - 174,6
      Фа-диез, соль-бемоль - 185,0
      Соль - 196,0
      Соль-диез, ля-бемоль - 207,7
      Ля - 220,0
      Ля-диез, си-бемоль - 233,1
      Си - 246,9
      Среднее до - 261,7
2-я октава
      До-диез, ре-бемоль - 277,2
      Ре - 293,7
      Ре-диез, ми-бемоль - 311,1
      Ми - 329,6
      Фа - 349,2
      Фа-диез, соль-бемоль - 370,0
      Соль - 392,0
      Соль-диез, ля-бемоль - 415,3
      Ля - 440,0
      Ля-диез, си-бемоль - 466,2
      Си - 493,9
      Среднее до - 523,3
```


Этот способ имеет существенный недостаток. На разных машинах высота звука будет «плавать», так как быстродействие ПП меняется от машины к машине из-за разброса параметров элементной базы. Устранить данный недостаток можно, «собирая» нужные частоты из набора стандартных. Сборка ведется по «И» (одна частота модулирует собой другую), что позволяет, управляя частотой программируемого таймера и комбинируя значения разрядов 8—12 регистра 177716, добиваться исполнения несложных мелодий.

Таблица 3

Разряды регистра 177710		Период тактовой частоты
2	1	
0	0	2 мкс
0	1	4 мкс
1	0	8 мкс
1	1	16 мкс

Данный способ дает стабильный звук, но его недостаток очевиден: крайне трудно получить требуемые ноты. Гораздо лучше использовать программируемый таймер в «чистом» виде в режиме прерываний. Достоинства такого способа: чистый стабильный звук, подбор частоты до единиц герц и легкость практической реализации. Действительно, раз можно плавно менять высоту звука и его длительность (это реализуется на другом таймере — сетевом), то уже можно исполнять довольно сложные мелодии. Для облегчения задачи приведем значения частот звучания нот двух октав в герцах.

```
        MOV     PC,R0           ;Вычислить адрес
                                программы обработки преры-
        ADD     #POP-.,R0       ;ваний от
                                программируемого таймера
        MOV     R0,@#304        ;Загрузить этот адрес в
                                вектор прерывания
        MOV     PC,R0           ;Вычислить адрес
                                программы обработки преры-
        ADD     #POPT-.,R0      ;ваний от сетевого
                                таймера
        MOV     R0,@#100        ;Загрузить этот адрес в
                                вектор прерывания
        MOV     PC,R0           ;Вычислить начальный
                                адрес массива с музы-
        ADD     #MUZON-.,R0     ;кой
        MOV     R0,ADRESS       ;Загрузить этот адрес в
                                ячейку ADRESS
        MOV     R0,EADRES       ;Вычислить и загрузить в
                                ячейку EADRES
        ADD     #<EP-MUZON>,EADRES ;конечный адрес
                                массива с музыкой
        BIS     #107,@#177710   ;Установить период
                                тактовой частоты програм-
        RTS     PC              ;мируемого таймера 16
                                мкс и разрешить
                                ;прерывания от него
        ;ПРОГРАММА ОБРАБОТКИ ПРЕРЫВАНИЯ ОТ СЕТЕВОГО
POPT:   .ENABLE LSB             ;ТАЙМЕРА
        DEC     (PC)+           ;Уменьшить на 1
                                содержимое ячейки COUNT
COUNT:  .WORD   50.             ;50 тиков - это 1
                                секунда (длительность ноты)
        BNE     1$              ;Если секунда не
                                истекла, то на выход
        MOV     @ADRESS,TON+2   ;Сыграть очередную ноту
                                из массива
        ADD     #2,ADRESS       ;Перейти на следующую
                                ноту
        CMP     EADRES,ADRESS   ;А не конец ли массива?
        BNE     2$
        SUB     #<EP-MUZON>,ADRESS ;Да . Вернемся к
                                началу
2$:     MOV     #50.,COUNT      ;1 секунда в COUNT
1$:     JMP     @#174612        ;Выход из прерывания
        ;ПРОГРАММА ОБРАБОТКИ ПРЕРЫВАНИЯ ОТ ПРОГРАМ-
POP:    .ENABLE LSB             ;МИРУЕМОГО ТАЙМЕРА
        TST     @#177714        ;Очистить регистр
                                текущего значения пр. тайм.
        MOV     R0,-(SP)        ;Сохранить в стеке R0
        MOV     #200,R0         ;Загрузить число 200 в
                                R0
        XOR     R0,@#177716     ;7 разряд если есть -
                                снять , и наоборот
        MOV     (SP)+,R0        ;Восстановить R0
```

Пример программы, иллюстрирующей применение программируемого таймера:

```
        .
TON:    MOV     #0,@#177712     ;Загрузить текущее
                значение ячейки TON+2 в буферный регистр
                программируемого таймера
        RTI                     ;Возврат из прерывания
ADRESS: .WORD   0               ;Ячейка хранения
                начального адреса массива
EADRES: .WORD   0               ;Ячейка хранения
                конечного адреса массива
        ;ДАЛЕЕ СЛЕДУЕТ САМ МАССИВ С МУЗЫКОЙ
MUZON:  .WORD   478.            ;До первой октавы
        .WORD   426.            ;Ре первой октавы
        .WORD   379.            ;Ми первой октавы
        .WORD   358.            ;Фа первой октавы
        .WORD   319.            ;Соль первой октавы
        .WORD   284.            ;Ля первой октавы
        .WORD   253.            ;Си первой октавы
        .WORD   239.            ;Срединное до
        .WORD   213.            ;Ре второй октавы
        .WORD   190.            ;Ми второй октавы
        .WORD   179.            ;Фа второй октавы
        .WORD   159.            ;Соль второй октавы
        .WORD   142.            ;Ля второй октавы
        .WORD   127.            ;Си второй октавы
        .WORD   119.            ;До второй октавы
EP:                             ;Конец программы в ПП
```

Эта программа проигрывает указанные ноты по порядку в цикле. Значения чисел, со-

ответствующие реальным звуковым частотам, рассчитываются по формуле A/F, где A соответствует тактовой частоте программируемого таймера (для периода 16 мкс A=62500, для 8 мкс A=125000, для 4 мкс A=250000, для 2 мкс A=500000 (все значения десятичные), а F — частота, которую требуется получить. Обратите внимание на то, что буферный регистр программируемого таймера (177712) 12-, а не 16-разрядный! Об этом следует помнить при загрузке данных в регистр.

С. ФАДЕЕВ
Санкт-Петербургский государственный университет с. н. с. ИПМ им. М. В. Келдыша

«Африка», «Матрешки» и словарная работа

Мы продолжаем рассказ об оригинальных компьютерных играх, предназначенных для изучения лексики иностранных языков.

Игра «Африка» не совсем обычна, ее с успехом можно применять как на уроках иностранных языков, так и на занятиях по географии. В ходе ее учащийся знакомится с животным и растительным миром Африки, а также отрабатывает соответствующую лексику конкретного иностранного языка (слон, лев, леопард, бегемот, обезьяна; пальма, баобаб и др.). Путешествуя по Африке, учащийся может столкнуться с любым животным, поведение которого в игре, как правило, приближено к его поведению к реальной среде. Например, если путешественник неправильно определит название хищника или слишком долго будет раздумывать над ответом, то этот хищник (крокодил, лев, леопард и др.) проглотит человечка на экране.

Если же человечек встретится с вполне мирным животным, то в случае неправильного ответа это животное просто убегает. При правильном ответе животное великодушно позволяет себя сфотографировать и лишь затем удаляется. Включение в игру фотоаппарата не случайно, так как в конце ее, вернувшись из путешествия по Африке, учащийся может еще раз проконтролировать свое знание «африканской» лексики, показывая своим друзьям фотографии (слайд-фильм) и называя изображенных на них животных.

Алгоритм игры довольно прост, поэтому на его основе можно создать аналогичные игры для отработки названий животных и растений других континентов, отдельных стран, районов и местностей.

В основе компьютерных игр «Садовник» и «Огородник» также лежит типовой алгоритм (зрительная семантизация+письменный контроль), который направлен на усвоение лексики по темам «Фруктовые деревья и кустарники», «Ягоды», «Цветы», «Овощи». Учащийся имеет возможность «вырастить» сад, разбить цветник или огород на экране компьютера. При этом количество фруктовых деревьев и овощных растений напрямую зависит от правильности или неправильности ответов учащегося. Лучшим садовником или огородником будет тот, кто правильно усвоил всю соответствующую лексику. С помощью подобных игр можно устраивать среди учащихся соревнования типа «Кто быстрее вырастит самый большой сад (огород)».

«Времена года» — небольшая игра, знакомящая с названиями месяцев и времен года. В начале ее на экране появляется календарь с названиями месяцев, несколько деревьев и шустрый зайчик. С помощью клавиш управления курсором можно перемещать зайчика с одного месяца на другой. При этом окраска деревьев и самого зайчика меняется в зависимости от месяца и времени года: в декабре, январе и феврале деревья и зайчик — белые, в марте деревья в отдельных местах покрываются зелеными почками, а зайчик начинает линять (к белому цвету примешиваются серые пятна), в апреле — мае зеленый цвет на деревьях и серый на зайчике становятся преобладающими. В июне, июле и августе деревья полностью зеленые, а зайчик — серый, кроме того, под деревьями появляются цветы.

В сентябре часть листьев на деревьях окрашивается в желтый цвет, а зайчик сно-

ва начинает линять. В октябре и ноябре деревья становятся желто-красными, цветы исчезают, а зайчик снова становится почти белым. Таким образом, учащийся может просмотреть все месяцы как по отдельности, так и по временам года.

В контролирующем режиме учащийся должен сам написать на календаре под номером месяца его название. Если ответ правилен, то на экране ярко светит солнце, а с окраской деревьев и зайчика происходят описанные выше изменения. Если же учащийся вводит неправильный ответ, то начинается дождь (зимой — снег), зайчик прячется под куст.

Есть и другая версия, в которой при правильном ответе на дереве появляется зеленый лист, при первом неправильном ответе — желтый, при втором — красный, а при третьем желтые и красные листья опадают. Таким образом, учащийся, усвоивший все названия месяцев и времен года, получит полностью зеленое дерево (лето). Учащийся, допустивший в ответах те или иные ошибки, получит либо весеннее (преобладание зеленого цвета), либо осеннее (преобладание желтого и красного) дерево. Если же лексика совершенно не усвоена, то на экране окажется дерево, покрытое снегом.

Как мы уже отмечали выше, данная игра наиболее логична при изучении названий месяцев и времен года, однако ее можно использовать в качестве игрового фона и при изучении любой другой лексики.

До сих пор мы рассматривали лишь компьютерные игры, направленные на зрительную семантизацию и контроль усвоения лексики. Однако лексические игры этим, конечно, не исчерпываются. Нами разработана также серия компьютерных игр, предназначенных для работы с абстрактной лексикой, синонимами и антонимами.

Игра «Матрешки» — одна из них. Первая версия этой игры направлена на усвоение синонимов. В начале игры на экране появляется одна большая матрешка, на которой написано какое-либо слово. Затем матрешка раскрывается, и из нее выходит другая, меньших размеров, на которой написан синоним к слову на большой матрешке. Эта процедура продолжается до тех пор, пока не выйдут все матрешки-синонимы.

В контролирующем режиме все слова на малых матрешках исчезают, остается только основное слово на большой матрешке. Задача учащегося — самостоятельно написать синонимы к нему. При правильных ответах матрешки будут прятаться одна в другую. Количество оставшихся после игры матрешек свидетельствует о числе синонимов, не усвоенных учащимся.

Вторая версия игры предназначена для тренировки антонимов. Она основана на обратном алгоритме. В начале игры на экране вокруг большой матрешки расположены более мелкие разной расцветки и величины. Слова, написанные на них,— антонимы к слову большой матрешки. После ознакомления с ними можно перейти к собственно игре, которая заключается в том, что на экране остается лишь одна большая матрешка. Необходимо напечатать все антонимы к слову, написанному на ней. При правильных ответах матрешки-антонимы будут выходить одна из другой. Специальный счетчик указывает на число нераскрытых матрешек-антонимов.

Третья версия используется для отработки значений многозначных слов. В ней могут использоваться алгоритмы как первой, так и второй версий, только здесь на большой матрешке написано слово на иностранном языке, а на малых — переводы всех его значений на русский язык. Возможен и обратный вариант, когда на большой матрешке русское слово, а на малых — перевод его значений на иностранный язык.

Игра «Рыбалка» также предназначена для тренировки усвоения синонимов, антонимов и многозначных слов. Игровая среда представляет собой изображение водоема, в котором плавает большая рыба, на боку ее в случайном режиме генерируется то или иное слово. В обучающей ситуации изо рта рыбы периодически выходят пузырьки воздуха, в которых написан один из синонимов (антонимов, переводов) к слову на боку рыбы.

В контролирующем режиме мимо большой рыбы проплывают маленькие, на них также написаны слова, некоторые из которых являются синонимами (антонимами, переводами) к слову большой рыбы. Учащийся может руководить движением большой рыбы с помощью клавиш управления курсором. Его задача заключается в том, чтобы подвести большую рыбу к маленькой, на которой, по мнению учащегося, написан синоним (антоним, перевод). Затем необходимо нажать клавишу «Ввод» и, если ответ правилен, то большая рыба хватает маленькую. В противном случае рыба сама попадает на блесну рыболова.

Синонимические и антонимические ряды, значения многозначных слов отрабатываются в серии спортивных игр: «Футбол», «Баскетбол», «Хоккей». Это наиболее сложные (с точки зрения программирования) игры. Игровая среда представляет собой имитацию известных игр в футбол, баскетбол и хоккей. Играют либо две команды синонимов, либо две команды антонимов, либо команды двух многозначных слов. На спинах игроков вме-

сте с номерами написаны слова, среди которых могут быть как синонимы (антонимы, переводы), так и случайные слова. В момент начала игры мяч (шайба) находится у одного из игроков; учащемуся необходимо быстро оценить обстановку, найти другого игрока этой же команды, на спине которого написан синоним (антоним, перевод) к слову на спине игрока с мячом, и ввести с клавиатуры номер этого игрока. Если ответ верен, то мяч (шайба) переходит к этому игроку и тем самым приближается к воротам (кольцу) противоположной команды. После серии правильных ответов мяч (шайба) попадает в кольцо или ворота противника. В результате же неправильного ответа или его длительной задержки мяч (шайбу) перехватывают соперники.

В различных версиях игры учащийся может играть и за обе команды, и за одну. В последнем случае его противником становится компьютер, который, как известно, можно запрограммировать как на выигрыш, так и на проигрыш. Первый вариант будет предпочтителен для сильных учащихся, так как при условии безошибочной игры они все-таки смогут свести матч к ничьей. Второй же вариант целесообразнее использовать новичкам, так как в этом случае компьютер как бы подыгрывает учащемуся, и у последнего появляется шанс на победу в матче.

«Десант» — еще одна программа, позволяющая в игровой форме поработать с антонимами, синонимами и многозначными словами. В ходе игры с летящего по экрану самолета прыгают парашютисты, на куполах их парашютов написаны различные слова. На земле приготовлены специальные площадки (синонимы, антонимы или переводы к словам на парашютах). Манипулируя клавишами управления курсором, учащийся должен посадить каждого парашютиста на нужную площадку. В начале игры учащийся имеет возможность задать ее сложность путем выбора количества парашютистов, которые прыгают с самолета с небольшой задержкой и приземляются поэтому в разное время. Но если учащийся уверен в правильности выбранной площадки (синонима, антонима или перевода), то он может ускорить спуск конкретного парашютиста, нажав соответствующую клавишу. Кроме того, в начале игры можно выбрать общую скорость движения парашютистов (медленную, среднюю, быструю). В случае неправильного ответа парашютист не достигает площадки, так как его парашют зацепляется за вершину дерева. Снять парашютиста с дерева можно лишь тогда, когда будет правильно написан синоним (антоним, перевод) к слову на парашюте. Этот прием самокоррекции используется в конце игры.

Игра «Посадка» аналогична предыдущей с той лишь разницей, что здесь необходимо посадить на специальные «синонимичные», «антонимичные» или «переводные» площадки не парашютистов, а вертолеты.

«Светофор» — универсальная лексическая игра, в которой цвет сигнализирует о правильности или неправильности ответа. Правильному ответу соответствует зеленый свет на светофоре, неправильному — красный, желтый свет горит в режиме ожидания ответа. Игровой фон можно использовать при обучении любой лексике, но наиболее целесообразно отрабатывать на нем глаголы движения (идти, ехать, переходить, доходить, подходить, обходить и др.). В первом случае зеленый свет на светофоре загорается только тогда, когда учащийся правильно определит либо название предмета, либо перевод слова, либо антоним или синоним слова, появляющегося на противоположной стороне улицы. Учащийся должен перевести всех пешеходов через улицу только на зеленый свет.

Во втором случае человечек на экране совершает движения, раскрывающие значения того или иного глагола движения. Компьютер комментирует его действия, например: «Он переходит улицу». В контролирующем режиме комментарий осуществляется учащимся, проверяющим правильность ответов по светофору.

При семантизации глаголов типа «ехать» на игровом фоне движется какое-либо транспортное средство (автомобиль, автобус, троллейбус и др.).

В другой версии «Светофора» учащийся начинает движение через улицу (ввод ответа) по желтому сигналу. В случае правильного ответа на светофоре загорается зеленый свет, и пешеход благополучно пересекает улицу. При неправильном ответе на светофоре вспыхивает красный свет, раздается милицейский свисток (звуковой сигнал), а пешеход возвращается назад. При повторном нарушении пешеход обязан уплатить штраф (дополнительный изучающий режим), а злостный нарушитель может в конце концов стать участником дорожно-транспортного происшествия.

«Скорый поезд» — еще одна универсальная игра, которую можно использовать как в обучающем, так и в контролирующем режиме. В обучающем режиме игра используется для семантизации как конкретной (с помощью зрительной наглядности), так и абстрактной (с помощью перевода) лексики; презентации синонимических и анто-

нимических пар; демонстрации морфемного состава слов.

В первом случае по экрану движется товарный поезд, на вагонах которого условные графические изображения и письменные названия перевозимых грузов либо иностранное слово с русским переводом. Во втором — пары синонимов или антонимов, в третьем — слова, которые расчленены окошками на морфемы.

В контролирующем режиме картина несколько иная. В первом случае, когда контролируется знание конкретной лексики, учащийся видит только графические изображения грузов, при контроле абстрактной лексики — слово на иностранном языке. Если контролируется знание синонимов или антонимов, то на вагоне будет один из них. И наконец, в третьем случае в слове, расчленением на морфемы, будет отсутствовать одна из морфем.

Таким образом, учащийся должен успеть написать за время движения поезда либо название груза, либо перевод иностранного слова, либо антоним или синоним, либо отсутствующую морфему. Если он это сделать не успевает или дает неправильный ответ, то поезд, не останавливаясь, уходит. Затем появляется следующий поезд, но это уже не экспресс, а обычный пассажирский, и скорость у него соответственно ниже. У учащегося появляется, следовательно, больше времени на ответ. Если и на этот раз в ответе допущена ошибка, то следующий состав, который придет на смену ушедшему, будет еле-еле тащить старинный паровоз. В случае новой ошибки учащегося паровоз совсем остановится перед красным семафором и будет стоять до тех пор, пока не загорится зеленый свет. Учащийся в это время будет повторять лексику в обучающем режиме, после чего сможет возобновить работу в режиме контроля и, исправив ошибку, отправить поезд со станции. Таким образом, в игре используется три скоростных режима, и играющий может совершенствовать свои навыки от «паровоза» (начальный уровень) до «экспресса» (высший уровень).

Джозеф Вейценбаум — известный американский ученый, работающий в Массачусетском институте технологии. Его замечательная книга «Возможности вычислительных машин и человеческий разум», вышедшая в 1976 г. и переведенная у нас в 1982 г., посвящена самым разным вопросам, связанным с ролью компьютеров в современном обществе. В ней, например, впервые проанализирован феномен «хакеров», которых Вейценбаум сравнивает с персонажем романа «Игрок» Ф. М. Достоевского.

Ниже приведена реплика Дж. Вейценбаума, выражающая его неожиданную для многих точку зрения на проблему компьютеризации образования (журнал «Byte», 1984, № 6).

В США мы охвачены массовым заблуждением относительно обучения детей с помощью компьютеров. Убеждение в необходимости срочно вводить компьютеры в начальную и среднюю школу основано на нескольких предпосылках, из которых лишь одна верна. Верная предпосылка заключается в том, что весь мир стремительно заполняется компьютерами. Но отсюда делается вывод: для преуспевания в мире, заполненном компьютерами, все должны быть «компьютерно грамотными». Делается вывод, что высокий уровень компьютерной грамотности гарантирует хорошую работу, тогда как компьютерная безграмотность обрекает на прозябание на задворках грядущего информационного общества.

Я думаю, что многие представляют себе компьютерную грамотность как умение общаться с компьютером, управлять им и уметь правильно понимать выдаваемую информацию. Следовательно, компьютерная грамотность обычно сводится к знанию одного или двух языков программирования и знакомству с клавиатурой ЭВМ.

Другая иллюзия заключается в представлении, что изучение языков программирования — сродни другим видам обучения, к которым лучше всего приступать как можно раньше. Эта точка зрения порождает мощное давление на школы с целью ввести компьютеры в обучение как можно раньше — ввести их в учебные программы от детского сада до 12-летнего возраста.

Конечно, компьютеры начали и будут продолжать заполнять наше общество. Я проведу аналогию с устройствами, также широко распространенными в нашем обществе — с электрическими моторами. Несомненно, что в США гораздо больше электромоторов, чем людей. Почти у каждого есть целая куча электромоторов, которые не бросаются в глаза или просто невидимы. Они повсюду — в автомобилях, миксерах, пылесосах, даже в часах и точилках для карандашей. Однако не

нужно никакой «электромоторной грамотности», чтобы пользоваться этими устройствами.

Компьютеры тоже становятся все более невидимыми. Я уверен, что чем больше их будет, тем более невидимыми они станут. Мы столько говорим о них из-за их новизны, но как только мы привыкнем к ним, они отойдут на задний план. Сколько компьютерной практики нужно учащемуся? Конечно, не очень много. Но и студент, и профессионал должны пользоваться специальными инструментами, в которые при необходимости компьютер войдет как составная часть.

Акцент на изучение компьютерных языков ошибочен. Очевидно, что языки ЭВМ не похожи на естественные. Они скорее напоминают языки математики или физики. Их изучение требует определенного интеллектуального уровня, й когда вы его достигли, то можете быстро научиться программировать. Не стоит тратить на это слишком много сил в раннем возрасте.

Контраргумент, что в раннем возрасте нужно начинать с простых вещей, например, учить 8-летних детей Бейсику, абсолютно неверен. С педагогической точки зрения Бейсик — интеллектуальное уродство, которое нужно искоренить и ни в каком виде не использовать как основу для чего-либо.

Я пытаюсь доказать, что введение компьютеров в начальную и среднюю школу — это ошибка, основанная на ряде предпосылок. Наши школы и так находятся в тяжелом состоянии, и введение компьютеров сейчас — это в лучшем случае диверсия, причем — очень опасная диверсия.

Очень часто компьютеры используются в школах так же, как и во многих других социальных институтах: чтобы на бумаге перескочить через фундаментальную проблему, чтобы создать иллюзию того, что на эту проблему ведется наступление.

Если Джонни не умеет читать, а кто-то сделал программу, которая может чуть-чуть улучшить умение Джонни, то легче всего принести компьютер и усадить за него Джонни. Теперь не нужно выяснять, почему же Джонни не умеет читать. Иными словами, это делает не нужным проведение реформы школьной системы и, следовательно, того общества, которое приводит к кризису систему образования.

ДЖ. ВЕЙЦЕНБАУМ

М. ПЕТРУХИНА

Программа «Журнал учителя»

Всем учителям знакома работа, связанная с ведением классного журнала. Когда драгоценное время учителя затрачивается на переписывание несколько раз списка класса, проставление оценок, подведение итогов успеваемости и посещаемости.

Предлагаемая программа позволит избавить учителя от рутинной работы с журналом, а также сэкономить его время.

Программа имеет следующие режимы работы

1. Открыть новый журнал.
2. Вывеети всю страницу журнала на экран.
3. Опрос учащихся.
4. Вывести данные успеваемости и посещаемости конкретного учащегося.
5. Вывести фамилии учащихся, имеющих конкретную оценку или пропуск.
6. Обновить журнал данного класса.
7. Вызвать произвольный журнал для работы.
8. Дополнить список класса новыми фамилиями.
9. Закончить работу.

Головная программа системы осуществляет общее управление набором подпрограмм и файлами прямого доступа, содержащими пояснение для работы. Все данные о классе записываются в файл последовательного типа, содержащий в своем названии предмет, по которому ведется данный журнал. Например, имя файла для журнала 9А по химии может выглядеть так: HCIM-9A, по физике FIZ-9A.

Рассмотрим наиболее принципиальные режимы работы программы.

Открыть новый журнал.

Режим позволит завести журнал на любой класс и по любому предмету. Здесь необходимо ввести предмет, фамилию учителя, класс, учебный год, количество учащихся и список всех учащихся. Максимальное число вводимых фамилий — 36.

Вывести всю страницу журнала на экран.

Для облегчения работы с журналом на экране он выглядит почти так же, как обычный журнал.

Химия. Павлова М. А. 9 А. 1991—1992 гг. 30 чел.

Фамилия, число	3/4	9	15	21 к.	27	5/5	12 л.	17	23
Матвеев	Н	Н		4			5		
Гурьева		5		5			5	Н	Н
Петров			3	4		4	5		
Фомин	4			4			4		
Макова			5	5		Н	4		
Гусева	3			3	4		4		4
Ларионов		4		4			5		
Боркова				5			4		

Программа дает возможность перелистывать страницу, если весь список не умещается в одном кадре. По количеству проведенных занятий журнал рассчитан на 1 четверть.

Опрос учащихся.

Данный режим позволяет как выставлять оценки за урок, так и отмечать отсутствующих в классе. При вводе даты текущего занятия можно делать различные пометки: «К» — контрольная, «Л» — лабораторная и т. д.

Обновить журнал данного класса.

Этот режим используется в случае, когда необходимо размножить список данного класса, для того, чтобы завести журналы по другим предметам. В этом случае необходимо ввести новое имя файла, т. е. название того журнала, который будет создан на основе старого. Далее меняется заголовок журнала, т. е. вводится новый предмет и фамилия учителя. В результате формируется новый журнал со старым списком класса. Этот режим используется и в случае, когда журнал весь заполнен оценками. Необходимо ввести имя старого журнала. Происходит как бы стирание оценок.

В заключение хотелось бы отметить, что использование предложенной системы позволяет не только вести журнал класса в классическом понимании, но и автоматизировать процесс контроля за успеваемостью, проводить статистический анализ и давать характеристику тенденции успеваемости учащихся.

Если школа оснащена компьютерной сетью, то данная программа становится хорошим помощником всем учителям, классным руководителям и директору, дает возможность хранить информацию о каждом учащемся и получать ее, не выходя из кабинета.

Можно создать банк данных, состоящий из журналов прошлых лет, своеобразный архив. При этом необходимо выбрать единую форму кодирования данных, чтобы исходя из имени файла, было ясно, в каком году и по какому предмету использовался данный журнал. Например: FIZ 9A392 — журнал по физике 9А класса, III четверть 1992 года.

Программа написана на языке Бейсик (Бейсик ПК8020) и адаптирована к ПЭВМ «Корвет». Программу можно перенести на любой тип ПЭВМ, используемый в школе.

Литература

1. *Моряхин А.* Электронный журнал с весовой функцией оценки // Информатика и образование. 1991. № 4.

Н. Фроленко

ПМК на уроках информатики

Создание методики изучения информатики и вычислительной техники в школах, техникумах и училищах предполагает разработку и экспериментальную проверку методики изучения основных понятий информатики и вычислительной техники — таких, как информация, алгоритм, язык программирования, ЭВМ, средства НИТ.

В настоящее время не уделяется должного внимания единому методическому подходу к использованию ПМК и переходу от работы с ПМК к работе на ЭВМ.

Программируемые микрокалькуляторы на начальных этапах изучения информатики и вычислительной техники могут стать базой для подготовки учащихся к общению с микрокомпьютерами любого типа, так как их использование допускает формирование таких понятий, как информация, алгоритм, программа на ЭВМ и др.

Использование программируемых микрокалькуляторов на начальном этапе изучения информатики позволяет заложить прочный фундамент в знаниях учащихся, необходимый для дальнейшей работы на ЭВМ, осуществить пропедевтическую подготовку для изучения курсов основ информатики и вычислительной техники. Таким образом,

программируемые микрокалькуляторы могут явиться важной базой для подготовки к общению с ЭВМ.

Ознакомление учащихся с машинноориентированным языком ПМК может стать дидактически исходным этапом обучения учащихся информатике, так как после изучения программирования на ПМК может быть осуществлена демонстрация возможностей ПЭВМ в более короткие сроки и с большим успехом. Именно тогда учащиеся практически готовы воспринять многие детали и принципы работы микро-ЭВМ, программного управления.

Кроме того, знакомство с ПМК позволяет учащимся составить достаточно четкое представление о том, как происходит процесс обработки информации в любой ЭВМ.

Назовем возможные пути применения ПМК в качестве пропедевтического средства изучения курса основ информатики и вычислительной техники:

1) ПМК в качестве учебно-вспомогательного инструмента;

2) ПМК в качестве рабочего инструмента при экономических, технических, проектных и бытовых расчетах;

3) ПМК в качестве средства освоения вычислительной техники и знакомства с принципами программирования;

4) ПМК в качестве игрового средства для проведения досуга.

Опишем некоторые аспекты методики обучения учащихся работе на ПМК, как пропедевтики изучения программирования на ЭВМ.

С самого начала нами рекомендован показ вычислительных примеров по математике, которые можно сопровождать упрощенной записью программы вычислений. Для этого целесообразно использовать графопроектор. Работа ведется таким образом: на экран проецируется через графопроектор расчетный бланк. Первые для программы (последовательное нажатие клавиш ПМК) показаны на экране, остальные программы и ответы закрыты. Преподавателю остается сравнить ответы. Такие расчетные бланки (кодотранспаранты) нами выполнены для курса «Основы информатики и вычислительной техники» по всему материалу в арифметическом и программированном режимах.

Для изучения предметов «Математика» и «Основы информатики и вычислительной техники» используются также плакаты и таблицы, изготовленные учащимися под руководством преподавателя. На таблицах демонстрируется либо учебный материал, необходимый для запоминания (постоянно демонстрируемые таблицы), либо материал в поддержку изучаемой темы курса.

Для индивидуальной работы с учащимися (для проведения контрольных, самостоятельных, практических и зачетных работ) используются карточки-задания, дифференцированные по трудности (обычно до 30 вариантов), а также карточки-задания для индивидуальной работы учащихся, предназначенные для повторения и закрепления учебного материала. Их можно использовать и во внеклассной работе.

Предполагаемая учебная работа на ПМК (постановка задачи, алгоритмизация, составление программы и др.) упрощает изучение различных тем курса информатики и вычислительной техники, обучение программированию на ЭВМ.

Кроме того, в нашей практике использования ПМК в качестве средства пропедевтики изучения основ информатики и вычислительной техники сделана попытка наметить основные типы знаний, которые должны быть сформированы у учащихся при их работе на ПЭВМ. Это нашло свою реализацию в построении учебного курса по компьютерной грамотности, который предусматривает усвоение пяти типов знаний: учащиеся должны научиться подготавливать компьютер к работе, выбирать программу из библиотеки программ, загружать программу в компьютер, работать с программой, общаться с компьютерной системой в период работы программы.

Для этих целей нами была разработана и практически апробирована программа учебного назначения «Допуск к работе на персональном компьютере «Роботрон-1715». Эта программа включает следующие блоки:

1. Общие сведения (описания базовых устройств ПЭВМ «Роботрон-1715»).
2. Обучение включению ПЭВМ, загрузке, включению, технике безопасности при работе с ПЭВМ.
3. Описание возможностей и особенностей работы монитора.
4. Описание возможностей и особенностей работы на клавиатуре.
5. Описание возможностей и особенностей работы принтера.
6. Обучение вводу программы в ЭВМ.
7. Обучение пользованию основными командами языка программирования «Бейсик».
8. Обучение пользованию основными операторами языка программирования «Бейсик».
9. Обучение основным приемам составления программ на ЭВМ.

При обучении работе на ЭВМ нами также использовались графопроектор, таблицы,

плакаты, карточки-задания для повторения, изучения и закрепления материала, кодотранспаранты, программы и трафареты, написанные самими учащимися.

Как показывает экспериментальная проверка, обучение учащихся работе на ЭВМ после занятий с ПМК проводится гораздо свободнее и на более высоком уровне, что, по всей видимости, объясняется пропедевтической подготовкой, осуществляемой с помощью ПМК.

Освоение учащимися навыков работы на ПМК и ЭВМ позволяет выполнять им нужные в повседневной практике расчеты лабораторных работ по физике, химии, курсовому проектированию. При этом учащиеся самостоятельно должны решать, какую работу следует выполнять на ПМК, а какую переносить на ЭВМ. Это является основой компьютерной грамотности учащихся, что формирует культуру учебной деятельности, основу информационной культуры учащихся.

Контрольно-тестовые программы с таймером и журналом регистрации

При существующей системе обучения преподаватель тратит на опрос студента значительную часть учебного времени. Необходимость разработки контрольно-тестовых программ (КТП), облегчающих монотонный труд преподавателя и одновременно позволяющих за считанные минуты осуществлять текущий контроль знаний студента и определять степень готовности его к выполнению лабораторно-практических занятий, становится все более очевидной [1, 2, 3].

Нами разработана КТП с использованием подпрограмм-функций:

Randomize (FF%) — выбора случайного числа при генерации переменной FF%;

RND — выбора случайного числа с начальной генерацией;

STR$(FF%) — преобразования числовой переменной в литерную;

LEN(FIO$) — определения длины литерной величины FIO$;

TIME$ (чч:мм:сс) — учета рационального использования контрольного времени.

Эффективность тест-контроля зависит от четкой постановки вопросов и ответов. В программе предусмотрена возможность изменения вопросов и ответов, которые могут быть как альтернативными, так и конструируемыми, и оценка последних по 10-балльной шкале. Нами использовались конструируемые ответы-синонимы, побуждающие студента к размышлению и анализу.

Апробация КТП с таймером показала, что подбор оптимального времени для каждой темы и на каждый ответ — это существенный психологический фактор, влияющий на качество и эффективность учебной деятельности. В используемых нами КТП оптимальным является предоставление 20 с на один ответ, и соответственно 200 с контрольного времени на 10 вопросов. С целью повышения активности ввели учет рационального использования контрольного времени, за экономию которого на 10 с дополнительно к основному (т. е. набранному за правильные ответы) добавляется 1 премиальный балл, за каждые просроченные 10 с 1 балл снимается. Учет контрольного времени с начислением премиальных и штрафных баллов способствует улучшению ориентирования учащихся в изучаемом учебном материале, повышает их активность, сокращает время тестового опроса.

В КТП предусмотрена выдача на экран дисплея и на принтер перечня вопросов, на которые студент не сумел правильно ответить, что позволяет выявлять «меру трудности» тестовых вопросов и «узкие места» в теоретической подготовке студента, а также косвенно оценить методический уровень преподавания данной темы. Это позволяет своевременно корректировать план самостоятельной работы, определять перечень теоретических вопросов и контрольных задач для индивидуальной работы со студентом.

Для удобства работы по крупным разделам КТП объединены посредством управляющей программы «меню», предусмотрена также автоматическая запись информации (Ф. И. О. студента, шифр специальности, № варианта и академической группы, количество набранных баллов, использованное время, оценка и дата работы) в файл «журнал преподавателя».

В соответствии с учебной программой (ГУМУ — 3/4 от 08.07.88 г.) разработаны и внедрены в учебный процесс ряд КТП по аналитической химии — 9 химических

методов анализа и 10 — физико-химических).

При оценке эффективности и научно-методической обоснованности КТП необходимо учесть следующие параметры:

а) мера трудности — доля неправильных ответов, позволяющая оценить степень усвоенности учебного материала;

б) общее количество набранных баллов;

в) рациональное использование контрольного времени, определяемое количеством набранных премиальных и штрафных баллов;

г) изменение процентного соотношения аттестованных студентов;

д) результаты зачетно-экзаменационной сессии, отработанные методами математической статистики (программа *MAX TAT*).

Количественная оценка влияния каждого параметра на эффективность разработанных тест-программ проводится в конце учебного семестра.

Литература

1. *Алипов Н., Соколов А.* Организация контроля знаний // Информатика и образование. 1989. № 5.

2. *Белая И., Дубильт А., Егорышева О., Юрина И.* Экспертная система определения «белых пятен» в знаниях ученика // Информатика и образование. 1991. № 2.

3. *Ушакова О., Володин Ю., Мардашев Ю.* Программа «Допуск» // Информатика и образование. 1991. № 1.

Ш. НОРОВ, Т. НИЯЗХОНОВ,
М. МАДЖИДОВ

Игры для КОРВЕТА

На заре своего существования компьютеры играли только в очень серьезные игры — например, шахматы использовались для исследований в области искусственного интеллекта. Однако именно «несерьезные» игры совершили переворот в компьютерном мире, вызвав массовый спрос на модели типа «Синклер Спектрум» и приведя экзотическую дотоле машину во многие дома. Пренебрегать подобным могуществом неразумно, особенно если учесть, что любая игра чему-то учит.

Мы с кружковцами педучилища поставили себе цель: составить такие программы, которые понравились бы малышам, учащимся I—III классов, у которых кружковцы проходят практику. И кажется, у нас получилось! Эти восемь и многие другие программы были встречены с восторгом. Дети быстро находили ответы и просили придумывать новые программы.

Сегодня мы предлагаем их всем желающим — быть может, вы тоже захотите поиграть. Конечно, в их текстах сказывается специфика «Корвета», но можно переделать их и на другую ПЭВМ.

```
10 REM ЗАСТАВКА
20 CLS:PCLS
30 LOCATE ,,0
40 LINE(2,2)-(509,254),7,B
50 LINE(5,4)-(509,252),6,B
60 CIRCLE(255,108),120,7,,,4
70 CIRCLE(255,108),128,6,,,4
80 PAINT(255,80)
90 CIRCLE (255,108),120,1,,,4
100 LOCATE 25,7:PRINT" С К А З К А "
200 FOR I=1 TO 600:NEXT
210 PRINT" ИВАН против КАЩЕЯ
                    БЕССМЕРТНОГО"
220 FOR I=1 TO 180 :NEXT
230 PCLS:CLS:PRINT TAB(14)," Помогу
      тебе, ИВАН, вызволить ВАСИЛИСУ
    ПРЕКРАСНУЮ, сказала БАБА ЯГА.- По
 душе ты мне привелся. Да и от кащеева
  коварства много я страдала, уж очень
                хочется его проучить"
240 FOR I=1 TO 360:NEXT
250 PRINT:PRINT:PRINT
260 PRINT "Учти, что все надписи на
                    дверях НЕВЕРНЫЕ !"
270 FOR I=1 TO 180:NEXT
280 PRINT:PRINT:PRINT
290 PRINT TAB(15),"Бросил ИВАН клубок
        на землю. Покатился клубок, а
     ИВАН - за ним. И докатился он до
        КАЩЕЯ. Потребовал Иван у него
              ВАСИЛИСУ ПРЕКРАСНУЮ."
300 PRINT:PRINT:PRINT
310 FOR I=1 TO 360:NEXT
320 PRINT"Повел КАЩЕЙ ИВАНА в
          подземелье. Показал там ТРИ
 темницы, на дверях которых написано:"
330 PRINT"Темница 1 - здесь Василиса
                        Прекрасная"
340 PRINT "Темница 2 - темница
                      3 НЕ пустая"
350 PRINT "Темница 3 - здесь
                     Змей Горыныч"
360 PRINT:PRINT:PRINT
370 FOR I=1 TO 420:NEXT
380 PRINT"-Отпушу, ИВАН, с тобой
    ВАСИЛИСУ ПРЕКРАСНУЮ,если угадаешь,
      в какой она темнице. Покажешь на
       дверь, за которой Змей Горыныч,
            быть тебе им растерзанным"
390 PRINT"Покажешь на пустую темницу -
      быть тебе в ней узником до конца
                      дней своих."
400 PRINT:PRINT:PRINT
405 PRINT"З А Д У М А Л С Я
                      И В А Н ... "
```

```
410 PRINT"Ребята, посоветуйте ИВАНУ,
        на какую дверь ему показать"
420 INPUT"Введите номер темницы :  ";X
430 IF X=1 THEN 510
440 PRINT:PRINT:PRINT
470 IF X=2 THEN 490
480 PRINT " ИВАН НАВЕКИ ЗАТОЧЕН В
                              ТЕМНИЦУ."
486 END
490 PRINT"      ИВАН встретился с
                ВАСИЛИСОЙ ПРЕКРАСНОЙ"
500 END
510 CLS:PRINT " ИВАН растерзан
                        ЗМЕЕМ ГОРЫНЫЧЕМ"



10 LOCATE 1,1,0:CLS:PCLS
20 FOR I=1 TO 490 STEP 10:
                  CIRCLE(I,230),8,7,,,2
30 CIRCLE(I,30),8,7,,,2:NEXT:
                FOR Y=30 TO 230 STEP 10
40 CIRCLE(10,Y),8,7,,,2:
                  CIRCLE(490,Y),8,7,,,2
50 NEXT
60 LOCATE 24,3,1
70 PRINT"ДОМИК КУМА ТЫКВЫ"
80 LOCATE 4,4,1:PRINT"Кум ТЫКВА
  с самого детства мечтал иметь домик"
90 LOCATE 4,5,1:PRINT"Он с 15 лет
       каждый год покупал 1 кирпич."
100 LOCATE 4,6,1:PRINT"Его друг
      Виноградинка посчитал кирпичи у
                                  него."
110 LOCATE 4,7,1:PRINT" Их оказалось
       18, и он сказал, что это мало"
115 PRINT
120 LOCATE 4,8,1:PRINT"Кум ТЫКВА решил
          мало есть и много работать"
125 PRINT
130 LOCATE 4,9,1:PRINT"Теперь он
            покупал 4 кирпича в год"
140 LOCATE 4,10,1:PRINT"Когда стало
      118 кирпичей, он стал строить."
150 LOCATE 4,11,1:PRINT"И через 2 года
          поселился у себя в домике."
160 LOCATE 4,12,1:PRINT"Сколько лет
                    было КУМУ ТЫКВЕ,"
170 LOCATE 4,13,1:PRINT"когда он
          окончил строить домик ?"
180 LOCATE 4,14,1:INPUT A%
185 CLS
190 IF A%=60  THEN 500 ELSE 195
195 PRINT TAB(10),,
        " н е п р а в и л ь н о !!!"
200 IF A%<>60 THEN GOTO 220 ELSE
                                GOTO 210
210 PCLS:END
220 FOR I=1 TO 300:NEXT:GOTO 10
500 PRINT TAB(20),
          " П Р А В И Л Ь Н О  !!!"
510 FOR I=20 TO 2 STEP -2
520 POKE T,I:BEEP:NEXT
530 LINE(100,60)-(300,60)
540 LINE(200,20)-(100,60),7
550 LINE(200,20)-(300,60),7
560 LINE(140,80)-(260,150),7,B
570 CIRCLE(200,118),40,,,,1.4
580 CIRCLE(180,114),5,,,,1.4
590 CIRCLE(220,114),5,,,,1.4
600 CIRCLE(220,114),2
610 CIRCLE(200,125),3
620 CIRCLE(200,135),5,,,,2.5
630 LINE(100,60)-(300,180),7,B
640 PAINT(110,70),7
650 PAINT(150,40),7
660 CIRCLE(180,114),2


10 CLS:PRINT"И Г Р А  -
                    У Г А Д А Й-К А  !"
20 X=INT(RND(1)*10)
30 Y=INT(RND(1)*10)
40 PRINT"СКОЛЬКО ПОЛУЧИТСЯ, ЕСЛИ
                        СЛОЖИТЬ X И Y ?"
50 PRINT"X И Y задумала З В М "
60 LOCATE 40,10
70 INPUT Z
80 A=X+Y
90 IF A=Z THEN 110 ELSE PRINT"НЕТ"
100 GOTO 20
110 CLS
120 LOCATE 20,10:PRINT"УРА !  РЕШИЛ !"
130 END
140 CLS
150 A:ST
160 160
170 150
190 A:ST
200 RUN


5 PRINT"ЗАДАЧА О БАЛДЕ"
10 LOCATE 1,1,0:CLS:PCLS
20 FOR I=10 TO 490 STEP 10:
                  CIRCLE(I,230),8,7,,,2
30 CIRCLE(I,30),8,7,,,2:NEXT I:
                FOR Y=30 TO 230 STEP 10
40 CIRCLE(10,Y),8,7,,,2:
                  CIRCLE(490,Y),8,7,,,2
50 NEXT Y
60 LOCATE 24,3,1
70 PRINT" П О П  и работник Б А Л Д А"
80 LOCATE 4,4,1:PRINT"С хозяйством
     попа справляются 10 работников."
90 LOCATE 4,5,1:PRINT"Каждый работник
          в день съедает каравай хлеба"
100 LOCATE 4,6,1:PRINT"Поп принял
                      на работу Балду."
110 LOCATE 4,7,1:PRINT"Ел Балда за
       четверых и работал за семерых"
115 PRINT
120 LOCATE 4,8,1:PRINT"Поп прогнал
                    лишних работников."
125 PRINT
130 LOCATE 4,9,1:PRINT"Сколько
  караваев хлеба экономил поп в день?"
180 LOCATE 4,14,1:INPUT A%
185 CLS
190 IF A%=3 THEN 500 ELSE 195
195 PRINT TAB(10),,"НЕПРАВИЛЬНО !"
210 PCLS:END
220 FOR I=1 TO 300:NEXT I:GOTO 10
500 PRINT TAB(20),"ПРАВИЛЬНО !"
510 FOR I=20 TO 2 STEP -2
520 POKE T,I:BEEP:NEXT



10 LOCATE 1,1,0:CLS:PCLS
20 FOR I=10 TO 490 STEP 10:
                  CIRCLE(I,230),8,7,,,2
30 CIRCLE(I,30),8,7,,,2:NEXT I:
                FOR Y=30 TO 230 STEP 10
40 CIRCLE(10,Y),8,7,,,2:
                  CIRCLE(490,Y),8,7,,,2
```

```
50 NEXT Y
60 LOCATE 24,3,1
70 PRINT "СТАРИК  ХОТТАБЫЧ"
80 LOCATE 6,4:PRINT"ВОЗРАСТ СТАРИКА
        ХОТТАБЫЧА ЗАПИСЫВАЕТСЯ ЧИСЛОМ
                    С РАЗЛИ ЦИФРАМИ."
90 LOCATE 5,6:PRINT"ОБ ЭТОМ ЧИСЛЕ
                ИЗВЕСТНО СЛЕДУЮЩЕЕ:"
100 LOCATE 5,7:PRINT"1. Если первую
     и последнюю цифры зачеркнуть, то
   получится двузначное число, которое
   при сумме цифр, равной 13, является
                        наибольшим."
110 LOCATE 5,10:PRINT"2. Первая цифра
          больше последней в 4 раза."
115 PRINT
120 LOCATE 5,11:PRINT"Сколько старику
                    ХОТТАБЫЧУ лет ?"
125 PRINT
180 LOCATE 4,14,1:INPUT A%
185 CLS
190 IF A%=8942 THEN 500 ELSE 195
195 LOCATE 23,8:PRINT"НЕПРАВИЛЬНО !"
200 IF A%<>60 THEN GOTO 220 ELSE
                                GOTO 210
210 PCLS:END
220 FOR I=1 TO 300:NEXT I:GOTO 10
500 LOCATE 21,8:PRINT " ПРАВИЛЬНО !"
510 FOR I=20 TO 2 STEP -2
520 POKE T,I:BEEP:NEXT
```

```
10 LOCATE 1,1,0:CLS:PCLS
20 FOR I=10 TO 490 STEP 10:
                CIRCLE(I,230),8,7,,,2
30 CIRCLE(I,30),8,7,,,2:NEXT I:
              FOR Y=30 TO 230 STEP 10
40 CIRCLE(10,Y),8,7,,,2:
                CIRCLE(490,Y),8,7,,,2
50 NEXT Y
60 LOCATE 20,3,1:PRINT"СТОЙКИЙ
                  ОЛОВЯННЫЙ СОЛДАТИК"
70 LOCATE 4,4,1:PRINT"БЫЛО НА СВЕТЕ
            25 ОЛОВЯННЫХ СОЛДАТИКОВ."
80 LOCATE 4,5,1:PRINT"ИХ СДЕЛАЛИ
          ИЗ ОЛОВЯННОЙ ЛОЖКИ МАССОЙ"
90 LOCATE 4,6,1:PRINT"123 ГРАММА.
         24 СОЛДАТИКА БЫЛИ ОДИНАКОВЫ"
100 LOCATE 4,7,1:PRINT",А 25-Й
                  НЕ ТАКОЙ, КАК ВСЕ."
110 LOCATE 4,8,1:PRINT"ОН ОКАЗАЛСЯ
           ОДНОНОГИМ. ЕГО ОТЛИВАЛИ"
120 LOCATE 4,9,1:PRINT"ПОСЛЕДНИМ, И
                  ОЛОВА НЕ ХВАТИЛО."
130 LOCATE 4,10,1:PRINT"КАКОВА МАССА
            ПОСЛЕДНЕГО СОЛДАТИКА ?"
140 LOCATE 4,14,1:INPUT A%
150 CLS
160 IF A%=3 THEN 190 ELSE 170
170 LOCATE 4,14,1:PRINT
      "     НЕПРАВИЛЬНО    ":GOTO 130
180 END
190 LOCATE 4,14,1:PRINT"ПРАВИЛЬНО"
200 END
210 IF Z<>0 THEN 230
220 LET Z=1
230 PRINT"ВЫ РЕШИЛИ
                  ВЕРНО";K;"ПРИМЕРОВ,"
240 PRINT"НЕПРАВИЛЬНО";N-K;"ПРИМЕРОВ,"
250 PRINT"ВАША ОЦЕНКА ";Z
260 PRINT"УСТУПИТЕ МЕСТО
                  ДРУГОМУ УЧЕНИКУ !"
270 GOTO 30
280 END
```

```
10 REM ЗАСТАВКА
20 CLS:PCLS
30 LOCATE,,0
40 LINE(2,2)-(509,254),7,B
50 LINE(5,4)-(504,252),6,B
60 CIRCLE(255,108),128,7,,,4
70 CIRCLE(255,108),128,6,,,4
80 PAINT(255,80)
90 CIRCLE(255,108),120,1,,,4
100 LOCATE 20,7:PRINT"ИЗБУШКА НА
                  КУРЬИХ НОЖКАХ"
130 CIRCLE(155,200),8
150 CLS
160 LINE(100,30)-(415,50),3,BF
170 LINE(100,30)-(415,50),7,B
180 LOCATE 15,3:PRINT"ИГРОВАЯ
                  ПРОГРАММА"
190 FOR L=1 TO 550:NEXT L
195 PCLS
196 CLS
200 PRINT"КАК НАДО ПЕРЕЛОЖИТЬ ДВЕ
    ПАЛОЧКИ, ЧТОБЫ ИЗБУШКА ПОВЕРНУЛАСЬ
                  ДРУГОЙ СТОРОНОЙ ?"
210 LINE(150,0) - (100,100),7
220 LINE - (100,200),7
230 LINE - (300,200),7
240 LINE - (300,100),7
250 LINE - (250,0),7
260 LINE - (150,0),7
270 LINE(100,100)-(200,100),7
280 LINE - (250,0),7
290 LINE(200,100)-(200,200),7
300 LINE(150,200)-(150,270),7
310 LINE(130,250)-(170,250),7
320 LINE(250,200)-(250,270),7
330 LINE(230,250)-(270,250),7
340 LOCATE 39,13:PRINT" ХОТИТЕ
                  ПОСМОТРЕТЬ ОТВЕТ ?"
350 INPUT"ВВЕДИТЕ -1,ЕСЛИ - Да";A
360 IF A=1 THEN 500 ELSE 200
500 CLS:PCLS:LINE(150,0)-(100,100),7
510 LINE-(100,200),7
520 LINE-(300,200),7
530 LINE-(300,100),7
540 LINE-(250,0),7
550 LINE-(150,0),7
560 LINE(200,100)-(300,100),7
570 LINE(200,100)-(150,0),7
580 LINE(200,100)-(200,200),7
590 LINE(150,200)-(150,200),7
600 LINE(150,200)-(150,270),7
610 LINE(130,250)-(170,250),7
620 LINE(250,200)-(250,270),7
630 LINE(230,250)-(270,250),7
```

```
20 PRINT"                  Т  Е  С  Т"
30 PRINT"       КАКОЙ ХАРАКТЕР У ВАШЕЙ
                              СОБАКИ?"
70 PRINT"При  О Т В Е Т Е   Вы должны
  нажать клавишу <1>(ДА) или <2>(НЕТ)
                  и клавишу <ВК>"
75 FOR I=1 TO 2000:NEXT
90 FOR I=1 TO 310
100 NEXT:S=0
110 FOR I=1 TO 320
120 NEXT
130 GOSUB 1070
140 PRINT"В чужом городе Ваша собака
          не боится отходить от Вас ?"
```

```
150 PRINT:PRINT
160 PRINT"1.ДА 2.НЕТ"
170 PRINT:PRINT
180 GOSUB 1100
190 IF M=1 THEN S=S+0
200 IF M=2 THEN S=S+1
210 GOSUB 1070
220 PRINT"Вы часто ищете контакта
                          с собакой ?"
230 PRINT:PRINT:PRINT
240 PRINT"1.ДА 2.НЕТ"
250 PRINT:PRINT
260 GOSUB 1100
270 IF M=1 THEN S=S+1
280 IF M=2 THEN S=S+0
290 GOSUB 1070
300 PRINT"Ваша собака любознательна ?"
310 PRINT:PRINT:PRINT
320 PRINT"1.ДА 2.НЕТ"
330 PRINT:PRINT
340 GOSUB 1100
350 IF M=1 THEN S=S+1
360 IF M=2 THEN S=S+0
370 GOSUB 1070
379 GOSUB 1070
380 PRINT"Ваша собака спит рядом с
                      Вашей кроватью ?"
390 PRINT:PRINT:PRINT
400 PRINT "1.ДА 2.НЕТ"
410 PRINT:PRINT
420 GOSUB 1100
430 IF M=1 THEN S=S+0
440 IF M=2 THEN S=S+1
450 GOSUB 1070
460 PRINT"Собака любит сидеть
                    с Вами в кресле ?"
470 PRINT:PRINT:PRINT
480 PRINT"1.ДА 2.НЕТ"
490 PRINT:PRINT
500 GOSUB 1100
510 IF M=1 THEN S=S+1
520 IF M=2 THEN S=S+0
530 GOSUB 1070
540 PRINT"К чужим людям она относится
                              хорошо ?"
550 PRINT:PRINT:PRINT
560 PRINT" 1.ДА 2.НЕТ"
570 PRINT:PRINT
580 GOSUB 1100
590 IF M=1 THEN S=S+1
600 IF M=2 THEN S=S+0
610 GOSUB 1070
620 PRINT" Ваша собака выполняет
                     Ваши поручения ?"
630 PRINT:PRINT:PRINT
640 PRINT"1.ДА 2.НЕТ"
650 PRINT:PRINT
660 GOSUB 1100
670 IF M=1 THEN S=S+1
680 IF M=2 THEN S=S+0
690 GOSUB 1070
700 PRINT" Ваша собака напоминает
                              Вам о еде ?"
710 PRINT:PRINT:PRINT
720 PRINT"1.ДА 2.НЕТ"
730 PRINT:PRINT
740 GOSUB 1100
750 IF M=1 THEN S=S+1
760 IF M=2 THEN S=S+0
770 GOSUB 1070
780 PRINT"Разворчива ли она в еде ?"
790 PRINT:PRINT:PRINT
800 PRINT"1.ДА 2.НЕТ"
810 PRINT:PRINT
820 GOSUB 1100
830 IF M=1 THEN S=S+1
840 IF M=2 THEN S=S+0
850 GOSUB 1070
860 PRINT "Вы с ней часто играете?"
870 PRINT:PRINT:PRINT
880 PRINT"1.ДА 2.НЕТ"
890 PRINT:PRINT
900 GOSUB 1100
910 IF M=1 THEN S=S+1
920 IF M=2 THEN S=S+0
930 GOSUB 1070
940 PRINT"Она не ревнует Вас
                              к друзьям ?"
950 PRINT:PRINT:PRINT
960 PRINT"1.ДА 2.НЕТ"
970 PRINT:PRINT
980 GOSUB 1100
990 IF M=1 THEN S=S+1
1000 IF M=2 THEN S=S+0
1010 GOSUB 1070
1020 IF S>8 THEN 1040
1030 IF S<8 THEN 1050
1040 PRINT"ВАША СОБАКА ОЧЕНЬ ДОБРАЯ,
   ЛЮБОЗНАТЕЛЬНАЯ, НИКОГДА НЕ БРОСИТ":
      PRINT"ВАС В БЕДЕ, ОНА НАСТОЯЩИЙ
                                ДРУГ.":END
1050 PRINT"ВАША СОБАКА АГРЕССИВНА,
 ПЛОХО ОТНОСИТСЯ К ВАШИМ ДРУЗЬЯМ.":END
1060 GOTO 1120
1070 FOR I = 1 TO 30
1080 PRINT
1090 NEXT:RETURN
1100 INPUT "ВВЕДИТЕ КОД ОТВЕТА"; M
1110 RETURN
1120 END
```

Пишите нам, обменяемся опытом. Мы составили также программы для уроков алгебры и геометрии для X—XI классов, для уроков информатики. Наш адрес: *357600, Ессентуки, ул. Долина Роз, 7, педучилище, каб. ОИВТ.*

Г. МЕДВЕДЕВА

В. КИРЮХИН

Международная олимпиада школьников по информатике

С 12 по 21 июля 1992 г. в Бонне проходила IV международная олимпиада по информатике (МОИ-92). В ней приняло участие рекордное число команд — 45, в том числе сборная команда России, которая впервые была представлена самостоятельно.

Интерес к этой олимпиаде был необычным. По сравнению с прошлым годом в МОИ-92 появилось много новых стран-участниц. Особое внимание вызвало первое выступление представителей таких стран, как США, Республики Корея, Австралии, ЮАР. Не менее интересным представлялось выступление команд республик СССР — Украины, Беларуси, Эстонии, Латвии и Литвы.

В состав сборной команды России по результатам национальной олимпиады и отборочных туров вошли:

Сергей Иоффе — выпускник сш № 82 п. Черноголовка, Московская обл.;

Дмитрий Жуков и Евгений Кузнецов — выпускники СУНЦ МГУ;

Дмитрий Давыдок — выпускник сш. № 239, г. Санкт-Петербург;

Руководители команды — к. т. н. доцент МИФИ **В. М. Кирюхин** и к. ф.-м. н. ассистент СУНЦ МГУ **Е. А. Андреева.**

Традиционно МОИ-92 проходила в два тура длительностью пять часов. На каждом из них участникам олимпиады предлагалось по одной задаче из пакета, подготовленного научным комитетом. Отбор задач проводило международное жюри непосредственно перед началом тура. В результате голосования были выбраны следующие задачи.

Задача 1 тура «Острова в море»

Расположение островов в море представлено с помощью сетки размером N*N. Каждый остров обозначается символом «*» в узле этой сетки. Задача заключается в том. По закодированной информации о распределении островов по горизонталям и вертикалям восстановить карту морских островов. Для иллюстрации принципа кодирования рассмотрим следующую карту и соответствующие ей коды:

```
*   * *                 1 2
  * * *   *             3 1
*   *   *               1 1 1
  * * ** *              5
* *   *   *             2 1 1
      *                 1

1 1 4 22 1
1 2   3  2
1
```

Числа справа от карты на этом рисунке являются кодами и представляют порядок и размер групп островов в соответствующих горизонталях сетки. Например, цифры «1 2» в первой строке означают, что первая горизонталь содержит группу из одного острова, за которой следует группа из двух островов. Слева и справа от каждой группы островов расположено море произвольной протяженности. Аналогично, последовательность «1 1 1» в первом столбце под картой островов означает, что первая вертикаль содержит три группы островов, в каждой из которых один остров, и т. д.

Постановка задачи

Написать программу, которая выполняет следующие действия (шаги) до тех пор, пока данный входной файл, содержащий несколько независимых блоков информации, не будет прочитан полностью:

1. Считывает очередной блок информации из входного A 11 файла (структура этого файла приведена в примере 1) и отображает его на экране.

Каждый блок информации начинается с величины размера квадратной сетки, за ко-

торым следует кодовая информация о горизонтальном и вертикальном распределении островов. Код для каждой отдельной горизонтали и вертикали является отдельной строкой файла и представляет собой последовательность чисел, разделенных пробелами. Заканчивается каждая строка нулем. При этом сначала идет информация о всех горизонталях, а затем — о вертикалях.

2. Восстанавливает карту (или все карты в случае не единственного решения как в примере 4) и отображает ее (их) на экране.

3. Добавляет карту (карты) в конец выходного ASCII файла. Каждое пустое место в сетке должно быть представлено двумя пробелами, а каждый остров — символом «*» (звездочка и пробел). В случае неоднозначного решения различные карты должны быть отделены друг от друга пустой строкой. Если карту восстановить невозможно, в соответствующей строке выходного файла написать «no map». Решения, относящиеся к различным блокам информации входного файла, должны быть отделены в выходном файле строкой с надписью «next problem».

Технические ограничения:

1. N должно быть не меньше 1 и не больше 8.

2. Результирующая программа должна быть помещена в текстовый ASCII файл с именем «C:`101`DAY—1`413—PROG. xxx». Расширение .xxx для PASCAL — .PAS для C — .C для BASIC — .BAS, для LOGO — LOG.

3. Имя входного ASCII файла должно быть «C:`101`DAY—1`413—SEAS .IN».

Пример 1

```
6
1  2  0          строка для первой горизонтали
3  1  0
1  1  1  0
5  0
2  1  1  0
1  0
1  1  1  0          строка для первого столбца
1  2  0
4  0
2  3  0
2  0
1  2  0
```

Пример 2

Блок входной информации	Решение
4	Столбцы 1 2 3 4
0	Строка 1
1 0	Строка 2 *
2 0	Строка 3 * *
0	Строка 4
0	
1 0	
2 0	
0	

Пример 3

```
2
0
0
2  0
2  0
```

Для этих данных не существует карты.

Пример 4

```
2
1  0
1  0
1  0
1  0
1  0
```

Этим данным удовлетворяют две различные карты.

Задача II тура «Покорение вершины».

Клуб альпинистов состоит из P членов, с номерами от 1 до P. Каждый альпинист поднимается в гору с одной и той же скоростью, а скорость подъема не отличается от скорости спуска. Альпинист с номером i расходует C(i) единиц ресурсов в день как при подъеме, так и при спуске, и может нести в каждый момент времени не больше S(i) таких единиц.

Предполагается, что альпинист может достичь вершины за N дней при полной обеспеченности ресурсами как для подъема, так и для спуска. Гора может быть так высока, что один альпинист не может изначально нести все необходимые для подъема и спуска ресурсы. Поэтому группа альпинистов стартует в одном и том же месте и в одно и то же время, чтобы обеспечить восхождение. Альпинист может начать спускаться, не достигнув вершины, отдав при этом все ненужные ему для спуска ресурсы другим альпинистам, которые должны быть в состоянии их взять. Альпинисты не отдыхают в течение экспедиции. Задача состоит в составлении расписания восхождения для клуба альпинистов. По крайней мере один альпинист должен достичь вершины горы и все альпинисты, включенные в группу восхождения, должны возвратиться в начальную точку, израсходовав все ресурсы.

Постановка задачи

Написать программу, которая выполняет следующее:

1. Вводит с клавиатуры число дней N, требуемое для достижения вершины горы, число P членов клуба и числа S(i), C(i) для всех i от 1 до P.

Все входные данные целочисленные, вещественные данные вводиться не будут. Повторить запрос на ввод данных, не имеющих смысла.

2. Определяет расписание для восхождения, а также номера альпинистов a(1), a(2), ..., a(k), образующих группу для восхождения, и числа M(j) (для всех j от 1 до k),

которые обозначают колнчество единиц ресурсов, взятых каждым a(j) альпинистом на старте.

Сообщает, если нельзя составить такого расписания для заданиых N, S(i) и C(i).

3. Выводит следующую информацию на экран:

а) число k альпинистов, участвующих в восхождении (размер группы восхождения);

б) необходимое для этого общее количество единиц ресурсов;

в) номера участвующих в восхождении альпинистов a(1), a(2), ..., a(k);

г) для каждого a(j), какое количество единиц ресурсов он возьмет с собой со старта;

д) для каждого a(j) количество дней, через которое он начнет спускаться;

4. Найденное расписание должно быть близким к оптимальному. Расписание является оптимальным, если:

а) число участвующих в восхождении альпинистов, необходимое для достижения вершины одним из них, минимально.

б) среди всех расписаний для групп, удовлетворяющих условию а), общее количество единиц ресурсов, взятых с собой со старта,— минимально.

Технические ограничения

1. Поместите вашу результирующую программу в текстовый ASCII файл с именем «C:\101\DAY—2\422—PROG.xxx». Расширение .xxx для PASCAL — PAS, для C — .C, для BASIC — .BAS, для LOGO — .LOG.

2. Программа не должна воспринимать входные данные, если N<1 или N>100, а также P<1 или P>20.

Пример:

Возможен следующий диалог с вашей программой.

Число дней, необходимое для достижения вершины — 4
Число альпинистов в клубе — 5
Максимальный ресурс для альпиниста 1 — 7
Ежедиевный расход ресурса для альпиниста 1 — 1
Максимальный ресурс для альпиниста 2 — 8
Ежедневный расход ресурса для альпиниста 2 — 2
Максимальный ресурс для альпиниста 3 — 12
Ежедневный расход ресурса для альпиниста 3 — 2
Максимальный ресурс для альпиниста 4 — 15
Ежедневный расход ресурса для альпиниста 4 — 3
Максимальный ресурс для альпиниста 5 — 7
Ежедневный расход ресурса для альпиниста 5 — 1
2 альпиниста требуется для покорения вершины
10 единиц ресурса в целом для этого необходимо.
Взбираться будут альпинисты 1, 5.
Альпинист 1 возьмет 7 единиц ресурса и начнет спускаться через 4 дня.
Альпинист 5 возьмет 3 единицы ресурса и начнет спускаться через 1 день.
Хотите спланировать восхождение для другого альпинистского клуба? (Y/N) — N.
Число дней, необходимое для достижения вершины — 2
Число альпинистов в клубе — 1
Максимальный ресурс для альпиниста 1 — 3
Ежедневный расход ресурса для альпиниста 1 — 1
Восхождение невозможно.
Хотите спланировать восхождение для другого альпинистского клуба? (Y/N) — N.
До свидания.

Задачи этой олимпиады продолжили традицию предыдущих: опять были предложены комбинаторные задачи, причем обе основаны на использовании перебора с возвратом. Основная идея алгоритма решения задачи 1 тура заключается в последовательном заполнении карт, представленных в виде матриц N×N, по кодовой информации о расположении островов в горизонталях с последующей проверкой на соответствие информации о расположении островов по вертикалям. В общем случае работа алгоритма должна заканчиваться, когда будут построены все допустимые карты.

Наилучшим способом реализации этого алгоритма является перебор с возвратом, который можно представить в виде дерева поиска. На первом уровне этого дерева формируются вышеназванные матрицы, первые строки которых соответствуют кодовой информации о расположении островов в горизонталях и не противоречат кодовой информации о расположении островов в вертикалях. На втором уровне полученные выше матрицы доопределяются второй строкой на основе кодовой информации о второй горизонтали, причем они также должны не противоречить информации о расположении островов в вертикалях. Продолжая этот процесс дальше, на нижнем уровне получим полностью заполненные матрицы, соответствующие искомым картам.

При реализации перебора с возвратом осуществляется обход дерева решений в глубину, отсекая те ветви, которые противоречат исходной кодовой информации. Это позволяет значительно сократить объем

перебора, так как бессмысленно на каком-то этапе дальше достраивать карту, если информация в начальных строках заполняемой матрицы противоречит имеющейся информации об островах в вертикалях.

Если первая задача относиться к классу перечислительных комбинаторных задач, то вторая — по своему характеру является оптимизационной. Понятно, что оптимальное решение можно получить в этом случае только путем полного перебора всех допустимых расписаний восхождения альпинистов. Однако попытки решить ее таким способом даже для указанных в условии интервалов изменения величин N и P не смогут привести к желаемому результату, в чем читатель сможет сам убедиться путем не очень сложных оценок, поскольку задача является NP-полной. Единственным путем является сокращение полного перебора на основе использования приемлемых эвристик. В частности, такие эвристики могут строиться, исходя из следующих утверждений, вытекающих из условия задачи:

— вершины должен достичь только один альпинист;

— каждый альпинист должен на старте взять ресурсов как можно больше и передать то, что ему не нужно, как можно раньше;

— наиболее вероятными для включения в группу восхождения являются альпинисты, способные опуститься и подняться как можно выше.

Задачи обоих типов оказались для многих участников олимпиады вполне доступными. Об этом говорит тот факт, что 12 школьников в итоге набрали максимально возможное количество баллов — 200. Большая группа участников также справились с обеими задачами, но им были незначительно снижены оценки за небольшие неточности, допущенные в процессе решения.

Так, в первой задаче важно было определить, что при корректных исходных данных существует не единственное решение. Например, если при заданном каждая строка содержит «1 0», то количество восстановленных карт может быть N! Кроме того, важно было выделить случаи, когда отсутствие решения может быть определено не в результате работы вышеописанного алгоритма, а в результате анализа входных данных. В частности, такими случаями являются:

а) превышение в строке суммы требуемых символов «×» размера сетки N;

б) равенство суммы кодов трех групп островов в строке значению (N—1), что определяет отсутствие места в строке для разделительных пробелов;

в) в отличие суммы всех кодов по строкам от суммы всех кодов по столбцам.

Во второй задаче важно было четко сформулировать утверждения, на основе которых можно ограничить полный перебор. К тому же следовало отметить, что отсутствие возможности восхождения в некоторых случаях можно определить на основе логического анализа исходных данных. Например, это справедливо для случая, когда ни для одиого альпиниста не выполняется условие $S(j)/C(j) \geqslant N$. Что касается условия $C(j) < S(j)$, то справедливость его должна устанавливаться при проверке данных на корректность.

Международное жюри осуществляло оценку решений задач с учетом разбалловок, приведенных в таблицах 1 и 2 для первой

ТАБЛИЦА 1

ОЦЕНИВАЕМОЕ РЕШЕНИЕ	БАЛЛЫ
1. Чтение блока информации из входного файла и отображение его на экране	5
2. Обработка всех информационных блоков один за одним, пока входной файл не будет считан полностью	10
3. Восстановление одной карты для каждого блока информации (если решение существует) и отображение его на экране	35
4. Запись карты решения в выходной файл	5
5. Восстановление всех возможных карт (если существует несколько решений) и отображение их на экране	20
6. Запись корректно разделенных карт решений в выходной файл	10
7. Идентификация блоков информации, для которых решения не существует	5
8. Выполнение всех технических требований	10

ТАБЛИЦА 2

ОЦЕНИВАЕМОЕ РЕШЕНИЕ	БАЛЛЫ
1. Организация диалога с пользователем как указано в условии	10
2. Найдено решение для частного случая, когда все C(1)=1 и все S(1) равны между собой	20
3. Найдено правильное решение для общего случая	20
4. Найдено как можно близкое к оптимальному решение для общего случая	30
5. Определены ситуации, которые не имеют решения	10
6. Выполнены все технические ограничения	10

и второй задачи соответственно. В результате золотые медали получили 13 олимпийцев, набравших 198—200 баллов. Серебряные медали получили те, кто набрал

175—195 баллов. Бронзовыми медалями награждены школьники, получившие 123—172 балла.

Среди участников, награжденных золотыми медалями, трое — из Китая, по два школьника — из США, Таиланда и Швеции, по одному — из Венгрии, Вьетнама, Чехо-Словакии и Республики Кореи.

Российские школьники получили одну серебряную медаль (Дмитрий Жуков — 187 баллов) и три бронзовые медали (Сергей Иоффе — 170 баллов, Евгений Кузнецов — 165 баллов, Дмитрий Давыдок — 131 балл).

Наилучших результатов на этой олимпиаде, как и ожидалось, добились школьники из Китая. Они получили три золотые и одну серебряную медали, набрав в общей сложности 785 баллов. За ними идут команды Таиланда (743 балла), Швеции (731 балл) и Республики Корея (728 баллов).

Увеличение числа участников существенно обострило борьбу за призовые места. Отрадным является тот факт, что несмотря на существующие трудности, команда России подтвердила свой высокий авторитет. По числу завоеванных медалей наша команда заняла шестое место (652 балла), совсем немного проиграв хозяевам олимпиады, объединенной команде Германии (671 балл) и опередив очень сильные команды Чехо-Словакии, США, Болгарии, Аргентины, Польши, Румынии, Великобритании.

Российским ребятам немного не хватило, чтобы завоевать хотя бы одну золотую медаль. Но, к сожалению, фортуна была не так к ним благосклонна, как хотелось бы. Это лишний раз доказывает, что мелочей в соревнованиях такого уровня не бывает. Чтобы побеждать, необходимо иметь существенный запас прочности, чего им и не хватало. Проявились старые наши недостатки — отсутствие стабильности в выступлениях, что является следствием недостаточной практики программирования на компьютерах. Это, в свою очередь, определяется отсутствием компьютерных центров в России, способных обеспечить потенциальных олимпийцев необходимыми условиями подготовки в области программирования. Беда, что наши талантливые ребята все еще не имеют доступа в необходимом количестве к персональным компьютерам. Да и не везде есть квалифицированные специалисты, способные оказать им помощь в обучении технике программирования, а наверстать упущенное за непродолжительное время подготовительных сборов невозможно, так как необходимые навыки можно получить только в процессе достаточно длительной подготовки.

С учетом сказанного интересно также отметить результаты команд из стран ближнего зарубежья: Беларуси, Латвии, Литвы, Украины и Эстонии. Сильная команда Украины уступила команде России 31 балл. Вслед за ними идут команды Литвы и Беларуси. Если бы названные государства, как и прежде, выступали единой командой, то, набрав 742 балла, эта команда заняла бы третье место вслед за Китаем и Таиландом, и в ее копилке было бы четыре серебряные медали, как в прошлом году.

Следующая международная олимпиада по информатике состоится в Аргентине в октябре 1993 г. Международный комитет в основном сохранил правила проведения соревнования в будущем году. Изменения внесены лишь в рабочие системы программирования, в качестве которых утверждены Turbo Pascal v. 6.0, Borland C++, а также Quick Basic или GWBasic.

В заключение хотелось бы пожелать будущим олимпийцам успешной подготовки к следующей олимпиаде. Думаю, что они смогут учесть опыт своих предшественников и достойно представят нашу страну в Аргентине.

Т. ПОЛИЛОВА
С. Н. С. ИПМ им. М. В. Келдыша РАН, Москва

Конференция «Графикон-92»

В Москве с 28 сентября по 2 октября 1992 г. под девизом «Компьютерная графика в науке и искусстве» проходила II Международная конференция по компьютерной графике. Конференция была организована Московским научно-техническим и творческим обществом компьютерной графики «Графо», Институтом прикладной математики им. М. В. Келдыша РАН, Союзом архитекторов, Институтом физики высоких энергий (г. Протвино) при содействии американского общества компьютерной графики ACM SIGGRAPH и европейского общества Eurographics.

Конференция Графикон-92 призвана поддержать кооперацию и информационный обмен Восток — Запад. Она стоит в ряду крупных ежегодных конференций ACM SIGGRAF в США, EUROGRAPHICS в Европе, NICOGRAPH в Японии.

С приветствием к участникам конференции обратился президент Российской Академии Наук академик Ю. С. Осипов.

Россия и другие страны СНГ переживают сегодня трудные времена. Однако именно сейчас, как никогда ранее, всем нам необходимы новые контакты, идеи, подходы, информация, новый взгляд и новые горизонты. Наука, как и прежде, является силой, объединяющей народы и определяющей прогресс цивилизации.

Большой успех предыдущей конференции Графикон-91 и проведенной в этом году конференции придает новый импульс международному сотрудничеству в области компьютерных технологий.

Впервые на конференции такого рода проводился семинар по проблемам образования. Ассоциация «Информационные технологии в образовании» (ИНТО) с участием фирмы IBM организовала научно-методический семинар «Компьютерная графика в образовании» (ведущий — В. Н. Афанасьев), программа работы которого включала выступления специалистов и коллективов, активно внедряющих новые информационные технологии, в том числе компьютерную графику, в учебный процесс.

Наука и образование всегда шли рядом, дополняя и подпитывая друг друга новыми идеями, сотрудничеством в подготовке специалистов. В обществе, претендующим называться цивилизованным, не может быть разрыва между двумя этими сферами. Поэтому особенно важно обеспечить непосредственный контакт между учеными, работающими на передовых рубежах науки, и преподавателями высшей и средней школы.

Последнее время наметился опасный разрыв между уровнем научных разработок в области информатики и компьютерных технологий, выполняемых в научных центрах страны, и уровнем подготовки специалистов в высшей школе. Не менее тревожная ситуация существует и в преподавании информатики в средней школе: плохое оснащение школ и вузов современной вычислительной техникой и программными продуктами, недостаток квалифицированных преподавателей, слабая материальная база учреждений образования.

Конференция, позволяя оценить перспективы в развитии средств вычислительной техники и информатики, новые тенденции, поможет скорректировать учебные программы таким образом, чтобы учащиеся были в курсе идей сегодняшнего дня.

Компьютерная графика в последнее время становится все более заметным течением в информатике. Интерес специалистов вызван потребностью современной науки в развитии новой технологии взаимодействия человека и компьютера. Для многих исследователей стала очевидной ограниченность вербальной формы представления информации на фоне суперкомпьютеризации вычислений. Современные мощные компьютеры производят слишком много цифр, чтобы их можно было понять и осмыслить. Если же визуализировать получаемые данные в виде картинок, динамических образов, то исследователь получает возможность изучать не только статическое состояние, но и эволюцию моделируемых на компьютере процессов и физических явлений.

С появлением компьютерной графики восстанавливается некая «историческая справедливость». Буквенно-цифровая технология общения человека с компьютером определила «левополушарный» крен в научных исследованиях. Но ведь процесс творчества неразрывно связан с интуитивными, образными механизмами. Компьютерная графика, вос-

производящая на экране сложные образы, помогает подключить эти механизмы творческого мышления.

Компьютерная графика способна интенсифицировать процесс научного открытия во многих областях фундаментальной науки. Как отмечают некоторые профессионалы, наиболее сложные математические результаты удается вначале «увидеть», а потом уже оформить в виде логически непротиворечивого утверждения. В последние несколько лет сложилось даже новое направление — когнитивная компьютерная графика, которая исследует связь и влияние графических образов на процесс научного творчества. Цель этих исследований состоит в разработке таких форм представления информации и манипулирования ею, которые были бы непосредственно обращены именно к интуиции, воображению и образному мышлению человека.

Компьютерная графика активно начинает использоваться и в образовании при создании обучающих и развивающих компьютерных программ для школы. Интересно заметить, что массовое увлечение школьников компьютерами началось в тот период, когда появились разнообразные компьютерные игры. Хотим мы того или нет, но компьютер становится для молодого человека интересным собеседником, равноправным партнером в играх. Да и сами компьютерные игры, зачастую, уже не пустая трата времени, а новые и неожиданные ситуации, в которых игроку нужно принимать нетривиальные решения, подключать интуицию для поиска пути, решать логические задачи.

Многие из имеющих хождение компьютерных игр хорошо иллюстрированы. Красивые картинки пробуждают воображение ребенка, развивают у него образное мышление, а игра приобретает ярко выраженную эмоциональную окраску. Лишены перспективы сухие и скучные компьютерные учебные программы, которые только и делают, что выводят на экран «поучительные тексты». Еще совсем недавно программа, построенная по принципу «введи ответ — получи оценку», могла претендовать на внимание к себе. Сегодня этого явно недостаточно.

Учебные материалы можно представить уже не только в виде текстов, графиков или схем, а с помощью графических образов, с использованием движения, мультипликации. При этом решаются не только чисто иллюстративные или оформительские задачи.

Известно, что уровень психического развития ребенка, в особенности младшего школьного возраста, не позволяет ему свободно оперировать понятиями, представленными в вербальной форме. Для этого возраста характерно наглядно-образное мышление с опорой на зрительные образы как средство решения мыслительных задач. Для ребенка, как впрочем и для взрослого человека, важна эмоциональная окраска процесса обучения. Яркие иллюстрации, интересный сюжет создают сильную мотивацию к процессу обучения и способствуют лучшему восприятию материала.

На конференции были представлены работы нескольких отечественных коллективов, которые предлагают программные продукты для школ и вузов на уровне требований сегодняшнего дня. Хочется отметить высокий профессиональный уровень многих представленных работ. И это вселяет надежду. Образование в наше трудное время сумело сохранить свой высокий потенциал, который нужно теперь эффективно использовать.

Международный конкурс фирмы Borland

Корпорация Borland International наряду с такими признанными лидерами, как Microsoft Corp. и Apple Compiter Inc., занимает на рынке программных продктов ведущие позиции в области разработки СУБД и объектно-ориентированных языков программирования.

Поглощение корпорации Ashton-Tate и приобретение прав на семейство баз данных dBASE привело к тому, что фирме Borland в настоящее время принадлежит около 70 % рынка СУБД и огромное число пользователей Paradox и dBASE, которые с нетерпением ждут появления новых версий этих программных продуктов, в том числе — и под Windows.

В 1993 г. корпорация Борланд (США) и Ассоциация Групп Пользователей Борланд в СНГ (БорАГ) проводят Третий Международный конкурс на лучшую разработку программных продуктов, выполненную с помощью инструментальных средств Борланд.

Название конкурса: «Борланд-Контест-93».

Цель конкурса: популяризация разработок фирмы, поиск новых идей и подходов в программировании и использовании программного обеспечения, стимулирование программистов, которые используют продукцию Борланд.

Девиз конкурса: «Интеллект + программирование = успех».

Условия участия в конкурсе «Борланд-Контест-93»:

1. Конкурс проводится для следующих групп программных продуктов:

средства для развития возможностей СУБД Paradox и dBASE, электронных таблиц Quattro Pro;

программные средства для работы под Windows;

объектно-ориентированные библиотеки классов и инструментальные средства для программистов-разработчиков;

обучающие программы;

компьютерные игры.

2. Итоги конкурса подводятся в двух возрастных категориях по отдельности:

«Борланд-Юниор» — школьники, учащиеся и студенты до 21 года;

«Борланд-Про» — профессионалы (любого возраста).

Любой участник может предложить свою программу в более «старшую» категорию, но не наоборот. Любой участник может предложить на конкурс не более одного продукта в каждой возрастной категории и в каждой группе программных продуктов.

Участники конкурса, не являющиеся членами БорАГ, должны оплатить организационный взнос в размере 100 руб. и приложить к сопроводительной открытке подтвердающий факт оплаты документ (р/с 700193 в Трансэкспобанке, корр. счет 161829 в РКЦ ГУ ЦБ РФ г. Москвы, МФО 201791).

Принятыми для участия в конкурсе считаются программы, направленные в адрес Оргкомитета с пометкой «Борланд-Контест-93» с 1 декабря 1992 до 31 января 1993 года (по почтовому штемпелю). Итоги конкурса будут подведены при участии специалистов Борланд и БорАГ и объявлены в июне 1993 г. на очередном Борланд-семинаре представителями фирмы Борланд.

За дополнительной информацией обращаться в Оргкомитет БорАГ: Москва, 125130, 2-й Новоподмосковский пер., д. 4
телефон (095) 150-92-01 (дем. зал), телекс 413560, телетайп 207321, телефакс 9430059

Закон Российской Федерации о правовой охране программ для электронных вычислительных машин и баз данных

Глава 1. Общие положения

Статья 1. Основные понятия

1. Основные понятия, применяемые в настоящем Законе:

программа для ЭВМ — это объективная форма представления совокупности данных и команд, предназначенных для функционирования электронных вычислительных машин (ЭВМ) и других компьютерных устройств с целью получения определенного результата. Под программой для ЭВМ подразумеваются также подготовительные материалы, полученные в ходе ее разработки, и порождаемые ею аудиовизуальные отображения;

база данных — это объективная форма представления и организации совокупности данных (например: статей, расчетов), систематизированных таким образом, чтобы эти данные могли быть найдены и обработаны с помощью ЭВМ;

адаптация программы для ЭВМ или базы данных — это внесение изменений, осуществляемых исключительно в целях обеспечения функционирования программы для ЭВМ или базы данных на конкретных технических средствах пользователя или под управлением конкретных программ пользователя;

модификация (переработка) программы для ЭВМ или базы данных — это любые их изменения, не являющиеся адаптацией;

декомпилирование программы для ЭВМ — это технический прием, включающий преобразование объектного кода в исходный текст в целях изучения структуры и кодирования программы для ЭВМ;

воспроизведение программы для ЭВМ или базы данных — это изготовление одного или более экземпляров программы для ЭВМ или базы данных в любой материальной форме, а также их запись в память ЭВМ;

распространение программы для ЭВМ или базы данных — это предоставление доступа к воспроизведенной в любой материальной форме программе для ЭВМ или базе данных, в том числе сетевыми и иными способами, а также путем продажи, проката сдачи внаем, предоставления взаймы, включая импорт для любой из этих целей;

выпуск в свет (опубликование) программы для ЭВМ или базы данных — это предоставление экземпляров программы для ЭВМ или базы данных с согласия автора неопределенному кругу лиц (в том числе путем записи в память ЭВМ и выпуска печатного текста) при условии, что количество таких экземпляров должно удовлетворять потребности этого круга лиц, принимая во внимание характер указанных произведений;

использование программы для ЭВМ или базы данных — это выпуск в свет, воспроизведение, распространение и иные действия по их введению в хозяйственный оборот (в том числе в модицированной форме). Не признается использованием программы для ЭВМ или базы данных передача средствами массовой информации сообщений о выпущенной в свет программе для ЭВМ или базе данных.

2. Под правообладателем в настоящем Законе понимается автор, его наследник, а также любое физическое или юридическое лицо, которое обладает исключительными имущественными правами, полученными в силу закона или договора.

Статья 2. Отношения, регулируемые настоящим Законом

1. Настоящим Законом и принимаемыми на его основе законодательными актами республик в составе Российской Федерации регулируются отношения, связанные с созданием, правовой охраной и использованием программ для ЭВМ и баз данных.

2. Программы для ЭВМ и базы данных относятся настоящим Законом к объектам авторского права. Программам для ЭВМ предоставляется правовая охрана как произведениям литературы, а базам данных — как сборникам.

Статья 3. Объект правовой охраны

1. Авторское право распространяется на любые программы для ЭВМ и базы данных, как выпущенные, так и не выпущенные в свет, представленные в объективной форме, независимо от их материального носителя, назначения и достоинства.

2. Авторское право распространяется на программы для ЭВМ и базы данных, являющиеся результатом творческой деятельности автора. Творческий характер деятельности автора предполагается до тех пор, пока не доказано обратное.

3. Предоставляемая настоящим Законом правовая охрана распространяется на все виды программ для ЭВМ (в том числе на операционные системы и программные комплексы), которые могут быть выражены на любом языке и в любой форме, включая исходный текст и объектный код.

4. Предоставляемая настоящим Законом правовая охрана распространяется на базы данных, представляющие собой результат творческого труда по подбору и организации данных. Базы данных охраняются независимо от того, являются ли данные, на которых они основаны или которые они включают, объектами авторского права.

5. Предоставляемая настоящим Законом правовая охрана не распространяется на идеи и принципы, лежащие в основе программы для ЭВМ или базы данных или какого-либо их элемента, в том числе на идеи и принципы организации интерфейса и алгоритма, а также языки программирования.

6. Авторское право на программы для ЭВМ и базы данных не связано с правом собственности на их материальный носитель. Любая передача прав на материальный носитель не влечет за собой передачи каких-либо прав на программы для ЭВМ и базы данных.

Статья 4. Условия признания авторского права

1. Авторское право на программу для ЭВМ или базу данных возникает в силу их создания. Для признания и осуществления авторского права на программу для ЭВМ или базу данных не требуется депонирования, регистрации или соблюдения иных формальностей.

2. Правообладатель для оповещения о своих правах может, начиная с первого выпуска в свет программы для ЭВМ или базы данных, использовать знак охраны авторского права, состоящий из трех элементов:

буквы С в окружности или в круглых скобках;

наименования (имени) правообладателя;

года первого выпуска программы для ЭВМ или базы данных в свет.

Статья 5. Авторское право на базу данных

1. Авторское право на базу данных, состоящую из материалов, не являющихся объектами авторского права, принадлежит лицам, создавшим базу данных.

2. Авторское право на базу данных признается при условии соблюдения авторского права на каждое из произведений, включенных в эту базу данных.

3. Авторское право на каждое из произведений, включенных в базу данных, сохраняется. Эти произведения могут использоваться независимо от такой базы данных.

4. Авторское право на базу данных не препятствует другим лицам осуществлять самостоятельный подбор и организацию произведений и материалов, входящих в эту базу данных.

Статья 6. Срок действия авторского права

1. Авторское право действует с момента создания программы для ЭВМ или базы данных в течение всей жизни автора и 50 лет после его смерти, считая с 1 января года, следующего за годом смерти автора.

2. Срок окончания авторского права на программу для ЭВМ и базу данных, созданные в соавторстве, исчисляется со времени смерти последнего автора, пережившего других соавторов.

3. Авторское право на программу для ЭВМ или базу данных, выпущенные анонимно или под псевдонимом, действует с момента их выпуска в свет в течение 50 лет. Если автор программы для ЭВМ или базы данных, выпущенных в свет анонимно или под псевдонимом, раскроет свою личность в течение указанного срока или принятый автором псевдоним не оставляет сомнений в его личности, то применяется срок охраны, предусмотренный пунктом 1 данной статьи.

4. Личные права автора на программу для ЭВМ или базу данных охраняются бессрочно.

Статья 7. Сфера действия настоящего Закона

Авторское право на программу для ЭВМ или базу данных, впервые выпущенные в свет на территории Российской Федерации либо не выпущенные в свет, но находящиеся на ее территории в какой-либо объективной форме, действует на территории Российской Федерации. Оно признается за автором, его наследниками или иными правопреемниками автора независимо от гражданства.

Авторское право признается также за гражданами Российской Федерации, программа для ЭВМ или база данных которых выпущена в свет или находится в какой-либо объективной форме на территории иностранного государства, или за их правопреемниками.

За другими лицами авторское право на программу для ЭВМ или базу данных, впервые выпущенные в свет или находящиеся в какой-либо объективной форме на территории иностранного государства, признается в соответствии с международными договорами Российской Федерации.

Глава 2. Исключительные авторские права

Статья 8. Авторство

1. Автором программы для ЭВМ или базы данных признается физическое лицо, в результате творческой деятельности которого они созданы.

Если программа для ЭВМ или база данных созданы совместной творческой деятельностью двух и более физических лиц, то независимо от того, состоит ли программа для ЭВМ или база данных из частей, каждая из которых имеет самостоятельное значение, или является неделимой, каждое из этих лиц признается автором такой программы для ЭВМ или базы данных.

2. В случае, если части программы для ЭВМ или

базы данных имеют самостоятельное значение, каждый из авторов имеет право авторства на созданную им часть.

Статья 9. Личные права

Автору программы для ЭВМ или базы данных независимо от его имущественных прав принадлежат следующие личные права:

право авторства — то есть право считаться автором программы для ЭВМ или базы данных;

право на имя — то есть право определять форму указания имени автора в программе для ЭВМ или базе данных: под своим именем, под условным именем (псевдонимом) или анонимно;

право на неприкосновенность (целостность) — то есть право на защиту как самой программы для ЭВМ или базы данных, так и их названий от всякого рода искажений или иных посягательств, способных нанести ущерб чести и достоинству автора.

Статья 10. Имущественные права

Автору программы для ЭВМ или базы данных или иному правообладателю принадлежит исключительное право осуществлять и (или) разрешать осуществление следующих действий:

выпуск в свет программы для ЭВМ или базы данных;

воспроизведение программы для ЭВМ или базы данных (полное или частичное) в любой форме, любыми способами;

распространение программы для ЭВМ или базы данных;

модификацию программы для ЭВМ или базы данных, в том числе перевод программы для ЭВМ или базы данных с одного языка на другой, иное использование программы для ЭВМ или базы данных.

Статья 11. Передача имущественных прав

1. Имущественные права на программу для ЭВМ или базу данных могут быть переданы полностью или частично другим физическим или юридическим лицам по договору.

Договор заключается в письменной форме и должен устанавливать следующие существенные условия: объем и способы использования программы для ЭВМ или базы данных, порядок выплаты и размер вознаграждения, срок действия договора.

2. Имущественные права на программу для ЭВМ или базу данных переходят по наследству в установленном законом порядке.

Статья 12. Имущественные права на программу для ЭВМ или базу данных, созданные в порядке выполнения служебных обязанностей

1. Имущественные права на программу для ЭВМ или базу данных, созданные в порядке выполнения служебных обязанностей или по заданию работодателя, принадлежат работодателю, если в договоре между ним и автором не предусмотрено иное.

2. Порядок выплаты и размер вознаграждения устанавливаются договором между автором и работодателем.

Статья 13. Право на регистрацию

1. Правообладатель всех имущественных прав на программу для ЭВМ или базу данных непосредственно или через своего представителя в течение срока действия авторского права может по своему желанию зарегистрировать программу для ЭВМ или базу данных путем подачи заявки в Российское агентство по правовой охране программ для ЭВМ, баз данных и топологии интегральных микросхем (далее — Агентство).

2. Заявка на официальную регистрацию программы для ЭВМ или базы данных (далее — заявка на регистрацию) должна относиться к одной программе для ЭВМ или одной базе данных.

Заявка на регистрацию должна содержать:

заявление на официальную регистрацию программы для ЭВМ или базы данных с указанием правообладателя, а также автора, если он не отказался быть упомянутым в качестве такового, и их местонахождения (местожительства);

депонируемые материалы, идентифицирующие программу для ЭВМ или базу данных, включая реферат;

документ, подтверждающий уплату регистрационного сбора в установленном размере или основания для освобождения от уплаты регистрационного сбора, а также для уменьшения его размера.

Правила оформления заявки на регистрацию определяет Агентство.

3. После поступления заявки на регистрацию Агентство проверяет наличие необходимых документов и их соответствие требованиям, изложенным в пункте 2 данной статьи. При положительном результате проверки Агентство вносит программу для ЭВМ или базу данных соответственно в Реестр программ для ЭВМ или реестр баз данных, выдает заявителю свидетельство об официальной регистрации и публикует сведения о зарегистрированных программах для ЭВМ и базах данных в официальном бюллетене Агентства.

По запросу Агентства или по собственной инициативе заявитель вправе до публикации сведений в официальном бюллетене дополнять, уточнять и исправлять материалы заявки.

4. Порядок официальной регистрации, формы свидетельств об официальной регистрации, состав указываемых в них данных устанавливаются Агентством. Агентством также определяется перечень сведений, публикуемых в официальном бюллетене.

5. Договор о полной уступке всех имущественных прав на зарегистрированную программу для ЭВМ или базу данных подлежит регистрации в Агентстве.

Договоры о передаче имущественных прав на программу для ЭВМ или базу данных могут быть зарегистрированы в Агентстве по соглашению сторон.

6. Сведения, внесенные в Реестр программ для ЭВМ или Реестр баз данных, считаются достоверными до тех пор, пока не доказано обратное.

Ответственность за достоверность указанных сведений несет заявитель.

7. За осуществление действий, связанных с официальной регистрацией программ для ЭВМ и баз данных, договоров, и публикацию сведений взимаются регистрационные сборы.

Размеры, сроки уплаты регистрационных сборов, а также основания для освобождения от их уплаты

или уменьшения их размеров устанавливаются Правительством Российской Федерации.

Глава 3. Использование программ для ЭВМ и баз данных

Статья 14. Использование программы для ЭВМ или базы данных по договору с правообладателем

1. Использование программы для ЭВМ или базы данных третьими лицами (пользователями) осуществляется на основании договора с правообладателем, за исключением случаев, указанных в статье 16 настоящего Закона.

2. Договор на использование программы для ЭВМ или базы данных заключается в письменной форме.

3. При продаже и предоставлении массовым пользователям доступа к программам для ЭВМ и базам данных допускается применение особого порядка заключения договоров, например, путем изложения типовых условий договора на передаваемых экземплярах программ для ЭВМ и баз данных.

Статья 15. Свободное воспроизведение и адаптация программы для ЭВМ или базы данных

1. Лицо, правомерно владеющее экземпляром программы для ЭВМ или базы данных, вправе без получения дополнительного разрешения правообладателя осуществлять любые действия, связанные с функционированием программы для ЭВМ или базы данных в соответствии с ее назначением, в том числе запись и хранение в памяти ЭВМ, а также исправление явных ошибок. Запись и хранение в памяти ЭВМ допускаются в отношении одной ЭВМ или одного пользователя в сети, если иное не предусмотрено договором с правообладателем.

2. В изъятие из положения абзаца третьего статьи 10 настоящего Закона лицо, правомерно владеющее экземпляром программы для ЭВМ или базы данных, вправе без согласия правообладателя и без выплаты ему дополнительного вознаграждения;

осуществлять адаптацию программы для ЭВМ или базы данных;

изготавливать или поручать изготовление копии программы для ЭВМ или базы данных при условии, что эта копия предназначена только для архивных целей и при необходимости (в случае, когда оригинал программы для ЭВМ или базы данных утерян, уничтожен или стал непригодным для использования) для замены правомерно приобретенного экземпляра. При этом копия программы для ЭВМ или базы данных не может быть использована для иных целей и должна быть уничтожена в случае, если дальнейшее использование этой программы для ЭВМ или базы данных перестает быть правомерным.

3. Лицо, правомерно владеющее экземпляром программы для ЭВМ, вправе без согласия правообладателя и без выплаты дополнительного вознаграждения декомпилировать или поручать декомпилирование программы для ЭВМ с тем, чтобы изучать кодирование и структуру этой программы при следующих условиях:

информация, необходимая для взаимодействия иезависимо разработанной данным лицом программы для ЭВМ с другими программами, недоступна из других источников;

информация, полученная в результате декомпилирования, может использоваться лишь для организации взаимодействия независимо разработанной данным лицом программы для ЭВМ с другими программами, а не для составления новой программы для ЭВМ, по своему виду существенно схожей с декомпилируемой программой для ЭВМ или для осуществления любого другого действия, нарушающего авторское право;

декомпилирование осуществляется в отношении только тех частей программы для ЭВМ, которые необходимы для организации такого взаимодействия.

Статья 16. Свободная перепродажа экземпляра программы для ЭВМ или базы данных

Перепродажа или передача иным способом права собственности либо иных вещных прав на экземпляр программы для ЭВМ или базы данных после первой продажи или другой передачи права собственности на этот экземпляр допускается без согласия правообладателя и без выплаты ему дополнительного вознаграждения.

Глава 4. Защита прав

Статья 17. Нарушение авторского права. Контрафактные экземпляры программы для ЭВМ или базы данных

1. Физическое или юридическое лицо, которое не выполняет требований настоящего Закона в отношении исключительных прав правообладателей, в том числе ввозит в Российскую Федерацию экземпляры программы для ЭВМ или базы данных, изготовленные без разрешения правообладателей, является нарушителем авторского права.

3. Контрафактными являются также экземпляры охраняемой в Российской Федерации в соответствии с настоящим Законом программы для ЭВМ или базы данных, ввозимые в Российскую Федерацию из государства, в котором эта программа для ЭВМ или база данных никогда не охранялись или перестали охраняться законом.

Статья 18. Защита прав на программу для ЭВМ и базу данных

1. Автор программы для ЭВМ или базы данных и иные правообладатели вправе требовать:

признания прав;

восстановления положения, существовавшего до нарушения права, и прекращения действий, нарушающих право или создающих угрозу его нарушения;

возмещения причиненных убытков, в размер которых включается сумма доходов, неправомерно полученных правонарушителем;

выплаты нарушителем компенсации в определяемой по усмотрению суда, арбитражного или третейского суда сумме от 5000-кратного до 50 000-кратного установленного законом размера минимальной месячной оплаты труда в случаях нарушения с целью извлечения прибыли вместо возмещения убытков;

помимо возмещения убытков или выплаты ком-

пенсации по усмотрению суда или арбитражного суда может быть взыскан штраф в размере десяти процентов от суммы, присужденной судом или арбитражным судом в пользу истца, в доход республиканского бюджета Российской Федерации;

принятия иных предусмотренных законодательными актами мер, связанных с защитой их прав.

2. За защитой своего права правообладатели могут обратиться в суд арбитражных или третейский суд.

3. Суд или арбитражный суд может вынести решение о конфискации контрафактных экземпляров программы для ЭВМ или базы данных, а также материалов и оборудования, используемых для их воспроизведения, и об их уничтожении либо о передаче их в доход республиканского бюджета Российской Федерации либо истцу по его просьбе в счет возмещения убытков.

Статья 19. Арест контрафактных экземпляров программы для ЭВМ или базы данных

На экземпляры программ для ЭВМ или базы данных, изготовленные, вопроизведенные, распространенные, проданные, ввезенные или иным образом использованные либо предназначенные для использования в нарушение прав авторов программы ЭВМ или базы данных и иных правообладателей, может быть наложен арест в порядке, установленном законом.

Статья 20. Иные формы ответственности

Выпуск под своим именем чужой программы для ЭВМ или базы данных либо незаконное вопроизведение или распространение таких произведений влечет за собой уголовную ответственность в соответствии с законом.

Президент
Российской Федерации — Б. Ельцин

Москва, Дом Советов России
23 сентября 1992 года
№ 3523-1

ПОСТАНОВЛЕНИЕ
ВЕРХОВНОГО СОВЕТА РОССИЙСКОЙ ФЕДЕРАЦИИ

О порядке введения и действие Закона Российской Федерации «О правовой охране программ для электронных вычислительных машин и баз данных»

Верховный Совет Российской Федерации постановляет:

1. Ввести в действие Закон Российской Федерации «О правовой охране программ для электронных вычислительных машин и баз данных» со дня опубликования.

2. Впредь до приведения законодательства Российской Федерации в соответствие с Законом Российской Федерации «О правовой охране программ для электронных вычислительных машин и баз данных» оно применяется постольку, поскольку не противоречит указанному Закону.

3. Закон Российской Федерации «О правовой охране программ для электронных вычислительных машин и баз данных» распространяется на отношения, связанные с созданием и использованием программ для электронных вычислительных машин и баз данных, возникшие после введения в действие указанного Закона.

4. Установить, что до 1 января 1994 г. на территории Российской Федерации использование в научно-исследовательских, учебно-просветительских и личных целях без извлечения прибыли программ для электронных вычислительных машин и баз данных, выпущенных в свет (опубликованных) до вступления в силу указанного Закона, допускается без согласия автора (правообладателя) и без выплаты ему вознаграждения.

5. Правительству Российской Федерации:

до 31 декабря 1992 г. внести в установленном порядке на рассмотрение Верховного Совета Российской Федерации проекты законов Российской Федерации о внесении изменений и дополнений в Гражданский кодекс РСФСР, в Уголовный кодекс РСФСР, другие законодательные акты, связанные с вопросами правовой охраны программ для электроиных вычислительных машин и баз данных;

до 1 января 1993 г. привести решения Правительства Российской Федерации в соответствие с Законом Российской Федерации «О правовой охране программ для электронных вычислительных машин и баз данных»;

до 1 января 1993 г. обеспечить пересмотр и отмену министерствами, государственными комитетами, ведомствами и другими организациями Российской Федерации их нормативных актов, в том числе инструкций, противоречащих указанному Закону;

до 31 декабря 1992 г. обеспечить принятие нормативных актов в соответствии с указанным Законом;

предусмотреть в республиканском бюджете Российской Федерации, начиная с 1992 г., все необходимые статьи затрат в соответствии с указанным Законом.

6. Комитету Верховного Совета Российской Федерации по науке и народному образованию с участием Государственного патентного ведомства Российской Федерации обобщить практику применения указанного Закона и о результатах доложить Верховному Совету Российской Федерации до 1 января 1994 г.

Председатель Верховного Совета
Российской Федерации — Р. И. Хасбулатов

Москва, Дом Советов Росии
23 сентября 1992 года
№ 3524-1

ПОСТАНОВЛЕНИЕ
ВЕРХОВНОГО СОВЕТА РОССИЙСКОЙ ФЕДЕРАЦИИ

О повторном рассмотрении Закона Российской Федерации «О правовой охране программ для электронных вычислительных машин и баз данных»

Рассмотрев возвращенный Президентом Российской Федерации Закон Российской Федерации «О правовой охране программ для электронных вычислительных машин и баз данных», Верховный Совет Российской Федерации п о с т а н о в л я е т:

1. В соответствии с частью второй статьи 117 Конституции (Основного Закона) Российской Федерации повторно принять Закон Российской Федерации «О правовой охране программ для электронных вычислительных машин и баз данных» с предложенным изменением текста статьи 2 указанного Закона.

2. Внести соответствующие изменения в пункт 5 постановления Верховного Совета Российской Федерации от 14 мая 1992 года «О порядке введения в действие Закона Российской Федерации «О правовой охране программ для электронных вычислительных машин и баз данных».

Председатель Верховного Совета
Российской Федерации

Москва, Дом Советов России
23 сентября 1992 года
№ 3225-1

УВАЖАЕМЫЕ ЧИТАТЕЛИ НАШЕГО ЖУРНАЛА!

Благодарим вас за оказанное доверие — тираж журнала в первом полугодии составил 31 000 экз., несмотря на то, что в 1992 году было издано всего четыре номера и цена увеличена до 50 руб.

Со своей стороны, несмотря на отсутствие с марта 1992 г. производственного помещения, реорганизацию издательства и неблагоприятные экономические условия, редакция принимала все меры, чтобы возобновить в 1993 г. выпуск всех шести номеров. В настоящее время на деньги, собранные от подписки на первое полугодие 1993 г. редакция гарантирует выпуск 1-го, 2-го и 3-го номеров журнала.

В связи с этим для тех, кто не успел и не смог подписаться на эти номера:

заявки с указанием вашего адреса, номеров и количества необходимых экземпляров просим направлять по адресу: 117485, Москва, а/я 34. Журналы будут отправлены вам по почте наложенным платежом (почтовые расходы за счет заказчика).

С 1 марта 1993 г. начинается подписка на 2-ое полугодие 1993 г. на 4-й, 5-й и 6-ой номера журнала «Информатика и образование». К сожалению, чтобы выпустить эти три номера, редакция вынуждена, как и другие педагогические журналы, увеличить цену. Мы очень надеемся, что наши постоянные читатели получат всю интересующую их информацию, и журнал будет им полезен и интересен.

Для индивидуальных подписчиков индекс журнала 70423. Цена 120 руб. за один номер.

Для учреждений и организаций (по безналичному расчету) индекс журнала 73176. Цена 240 руб. за один номер.

Если у вас не принимают подписку по безналичному расчету, обращайтесь в ЦРПА «Роспечать» по телефону (095) 195-64-25.

Хотите быстро подготовить и качественно провести занятия по информатике ?

Вам поможет
Программно-методический комплекс

«Информатика»

В состав комплекса входят:

Для преподавателей

☞ Учебная программа, пояснительная записка, тематический план с указанием лабораторно-практических работ.

☞ Автоматизированная обучающая система «ЭСКОРТ». Необходима при подготовке и проведении занятий по информатике. Имеет 2 режима работы: обучение и зачет; 2 уровня обучения: простой и сложный.

Комплекс можно приобрести целиком или его отдельные составные части. Имеются версии для ДВК, УК-НЦ, УК-НЦ 01.01, КУВТ-86.

Наш адрес: 107005, Москва, Волховский пер., дом 11
Фирма «Колледж»
Телефон: 267-70-58

КОЛЛЕДЖ

◆ Поставка компьютерного класса «под ключ»

◆ Переналадка старых классов

◆ Изготовление мебели для школ и офисов

Для работы с учащимися

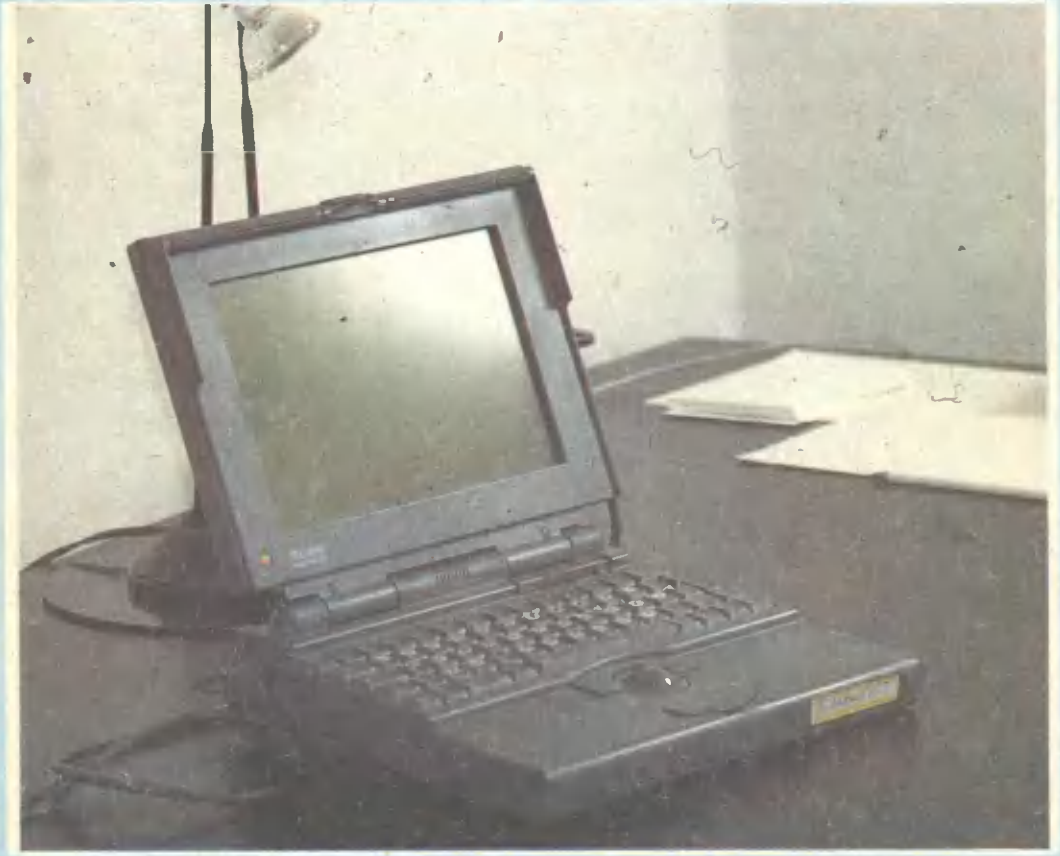
☞ Автоматизированная обучающая система «КОЛЛЕДЖ». Предназначена для обучения учащихся школ и средних специальных учебных заведений по курсу «ОИ и ВТ».
АОС «КОЛЛЕДЖ» охватывает темы: первый контакт с ЭВМ, представление информации в ЭВМ, основные принципы устройства и работы ЭВМ, операционная система, алгоритмизация и решение задач на ЭВМ, знакомство с программированием на BASIC.

☞ Сборник дидактических материалов для лабораторно-практических работ (по 30 вариантов карточек-заданий для проведения 20 работ).

☞ Пакет прикладных программ.

☞ Уроки с программируемым микрокалькулятором (с примерами и индивидуальными заданиями).

Цена 50 руб. по подписке
70423



ISSN 0234—0453. Информатика и образование, 1993, № 1, 1—128.