В. З. Аладьев,
Я. Г. Мартыненко,
В. Ф. Шиленко

ПЕРСОНАЛЬНЫЙ КОМПЬЮТЕР

ИСКРА 226

АРХИТЕКТУРА И ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ

Справочное руководство

УРЕ

В. З. АЛАДЬЕВ,
Я. Г. МАРТЫНЕНКО,
В. Ф. ШИЛЕНКО

ПЕРСОНАЛЬНЫЙ КОМПЬЮТЕР

АРХИТЕКТУРА И ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ

Справочное руководство

Под общей редакцией
В. Ф. Шиленко

Киев
Главная редакция
Украинской Советской Энциклопедии
имени М. П. Бажана
1988

ББК 32. 973. 2я2
А45

УДК1 + 681.3

Рецензент академик АН УССР В. И. СКУРИХИН
Научная редакция физики, математики и химии
Зав. редакцией Ж. Д. Гудзенко

Аладьев В. З. и др.

А45 Персональный компьютер „Искра-226". Архитектура и программное обеспечение: Справ. руководство / В. З. Аладьев, Б. Г. Мартыненко, В. Ф. Шиленко; Под общ. ред. В. Ф. Шиленко. – В пер. – К.: Глав. ред. УСЭ им. М. П. Бажана, 1988. – 152 с.: ил. –4.:Библиогр. – 62.: 70 к. 50 000 экз.

ISBN5–88500–004–2.

В справочном руководстве рассмотрены технические и программные средства ЭВМ „Искра–226", описаны операторы языка БЕЙСИК и особенности их выполнения. Даны прикладные программы, используемые в операционной среде БЕЙСИКа, в том числе программа А – BASIC, позволяющая расширить его выразительные средства.
Предназначено для специалистов, широкого круга читателей, желающих использовать в своей деятельности возможности программно-управляемых ЭВМ.

А $\frac{2404040000-006}{M\ 222\ (04)-88}$ Б 3–7–6–88

ББК 32. 973.2я2

ISBN 5–88500–004–2

ПРЕДИСЛОВИЕ

Бурное развитие той или иной сферы человеческой деятельности на первых порах, как правило, не ставит на повестку дня решение проблем оптимизационного характера в этой сфере. Идет нарастающее накопление результатов, экспериментальных и теоретических данных, в сферу деятельности вовлекается все больше специалистов разных профилей, администраторов и т. д. И только спустя какое-то время, в период своего становления актуальными становятся проблемы оптимизационного характера: как тот или иной вопрос решить именно оптимальным образом.

В полной мере это можно отнести и к такой важной сфере человеческой деятельности, как вычислительная техника (ВТ), развивающейся в настоящее время особенно бурно. На первых этапах развития ВТ не ставился целый ряд оптимизационных проблем, которым теперь уделяется много внимания. Правда, и тогда надо было решать задачи, связанные с повышением производительности ЭВМ, уменьшением их размеров, стоимости и т. д. Но сегодня на повестке дня одной из важнейших является проблема эффективного применения ВТ в народном хозяйстве [3–5,7].

Опыт использования ВТ во многих странах мира показал, что путей решения этой проблемы есть несколько в зависимости от ее детализации. Так, в процессе развития ВТ выделились три основных класса ЭВМ: универсальные, мини- и микро-ЭВМ, представители каждого из которых существенно отличаются между собой. Если раньше особенно стремительно развивался и расширялся класс универсальных ЭВМ, то теперь это в полной мере можно отнести и к классам мини- и микро-ЭВМ, а также персональным компьютерам (ПК). Вероятно, не имеет смысла выяснять здесь причины такого явления, поскольку оно детально обсуждалось на различных уровнях [6].

В нашей стране в последние годы эта тенденция также осталась весьма ощутимой, что подтверждается появлением ряда мини- и микро-ЭВМ разного применения (СМ ЭВМ, "Нева", "Искра", "Электроника") [6]. Среди них особое место занимает ЭВМ "Искра-226", прототипом для которой послужила хорошо известная ЭВМ "WANG – 2200".

Электронная вычислительная машина "Искра-226" ориентирована на проведение в диалоговом режиме оперативных плановых расчетов, работу с локальными банками данных в составе информационных и поисковых систем, решение в диалоговом режиме научно-технических, инженерных, статистических и оптимизационных задач. ЭВМ "Искра-226" может быть эффективно использована в сетях ЭВМ в качестве интеллектуального терминала или элементарной ЭВМ на низшем уровне сетевой обработки [7], а также в системе автоматизированных рабочих мест (АРМ).

Еще об одном моменте следует сказать особо. Порой очень трудно провести четкую грань между тем или иным классом ЭВМ. Поэтому в различных книгах, технической документации и других материалах по ЭВМ имеется немалая терминологическая путаница. На наш взгляд, "Искра-226" должна быть отнесена к классу ПК. Но чтобы избежать споров по этому вопросу, который для целей данной книги не является принципиальным, далее по тексту будем использовать термин просто ЭВМ либо ЭВМ "Искра-226", подразумевая полное техническое название "Машина вычислительная электронная программно-управляемая "Искра-226", данное ей разработчиками.

Необходимо сделать еще одно важное замечание. Несмотря на имеющийся в стране положительный опыт подготовки документации по программному сопровождению ЭВМ [1, 2], представляющей особый интерес для пользователей, техническая документация для ЭВМ "Искра-226" оставляет желать много лучшего. Составлена она в предположении определенного уровня подготовки пользователя, что, вообще говоря, не должно относиться к ЭВМ подобного класса. Поэтому в настоящей книге сделана попытка проложить тот мост, который поможет связать желания пользователя с возможностями ЭВМ "Искра-226" при минимальных затратах со стороны пользователя. Для успешного самостоятельного освоения ЭВМ "Искра-226" наряду с данной книгой следует пользоваться соответствующей документацией по программному обеспечению (ПО) ЭВМ [9–23].

Заметим еще, что приведенный ниже материал в равной мере относится также к модификациям ЭВМ "Искра-226" и комплексам программно-технических средств (КПТС) на ее основе: ЭВМ "Искра-226М", АРМ технолога-программиста, КПТС для системы обработки текстов (ЭВМ "Искра-226 СОТ").

СПИСОК СОКРАЩЕНИЙ

АВ – арифметическое выражение
АЛУ – арифметико-логическое устройство
АРМ – автоматизированное рабочее место
АС – алгебраическая система
АСОД – адаптируемая система обработки данных
АЦПУ – алфавитно-цифровое печатающее устройство
БД – база данных
БИС – большая интегральная схема
БОСГИ – блок отображения символьной и графической информации
ВТ – вычислительная техника
ЕС ЭВМ – единая система электронных вычислительных машин
ЗЧ – запоминающая часть
ИБ – интерфейсный блок
ИВВ – интерфейс ввода–вывода
ИДП – интерпретирующий диалоговый процессор
ИМС – интегральная микросхема
ИП – интерпретируемая программа
ИПС – интерактивная программная система
к-ЗЛФ – к-значная логическая функция
ККТП – комплект контрольных тест-программ
КП – клавишный пульт
КПР – канальный процессор
КПТС – комплекс программно-технических средств
КУ – канальное устройство
ЛН – логический номер
ЛУ – логическое условие
НГМД – накопитель на гибких магнитных дисках
НМД – накопитель на магнитных дисках
НМЛ – накопитель на магнитных лентах
НП – номер программы
НСЧ – непосредственный счет
ОП – оперативная память
ОС – однородная структура
ОСР – однородная структура с рефректерностью
ОУП – оперативная управляющая память
ПДС – параллельная динамическая система
ПЗ – признак защиты
ПЗУ – постоянное запоминающее устройство
ПК – персональный компьютер
ПМК – память микрокоманд
ПО – программное обеспечение
ПООС – последовательность однозначно определенных сумм
ПОП – программное обеспечение пользователя
ППП – пакет прикладных программ
ПрФ – признак файла
ПС – программная строка
ПСОИ – параллельная система обработки информации
ПФ – программный файл
РБ – регистр базовый
РК – регистр команд
РОН – регистр общего назначения

РП – регистр признаков
РР – регистр рабочий
РС – регистр сумматора
СВОП – сверхоперативная память
СК – схема контроля
СМ ЭВМ – система малых электронных вычислительных машин
СПВ – схема приема и выдачи информации
СПО – системное программное обеспечение
СС – схема согласования
СУ – схема управления
ТУВВ – таблица устройства ввода – вывода
ТУС – таблица устройств системы
УВВ – устройство ввода – вывода
УК – указатель каталога
УОП – управляемая оперативная память
УП – управляющая память
ФАУ – физический адрес устройства
ФД – файл данных
ЦП – центральный процессор

1. АРХИТЕКТУРА СЕМЕЙСТВА ЭВМ "ИСКРА-226"

1.1. Назначение и основные технические данные ЭВМ

Электронная вычислительная машина "Искра-226" поставляется пользователю в одном из шести исполнений, отличающихся своей комплектностью (табл. 1). Характеристика и технические данные каждого из устройств ЭВМ помещены в [8].

В зависимости от исполнения ЭВМ предназначена для:

проведения в диалоговом режиме оперативных плановых расчетов и выдачи по ним выходных печатных форм;

работы с локальными базами данных в составе информационно-поисковых и справочных систем;

решения в диалоговом режиме научно-технических, экономических и оптимизационных задач;

работы в сети телеобработки данных в качестве программируемого терминала СМ/ЕС ЭВМ;

автоматизации ряда процессов исследовательских, проектных, конструкторских и инженерных работ в СКБ, КБ, НИИ с представлением информации на дисплее и документированием результатов;

решения научно-технических задач при непосредственном участии пользователя (конструктора, инженера, научного работника) в процессе вычислений;

обработки информации в системах автоматизации научных исследований при непосредственном участии пользователя;

работы в качестве интеллектуального терминала.

Любое исполнение ЭВМ "Искра-226" обеспечивает выполнение следующих основных задач:

ввод программ и данных с пульта оператора;

редактирование программ;

вывод программ и данных на дисплей или печать;

телепроцессинг по рангу СГ (ГОСТ 18145–72) с ЕС и СМ ЭВМ;

интерпретацию команд программы, записанной в памяти.

Дополнительно (в зависимости от исполнения) ЭВМ позволяет осуществлять:

ввод – вывод системного программного обеспечения (СПО) с НМД, обмен информацией с НГМД (исполнения 1);

ввод – вывод СПО с НМД, обмен информацией с НМД (НГМД) (исполнение 2);

Таблица 1. Комплектность ЭВМ "Искра-226" в зависимости от исполнения

Номер пор.	Наименование устройств	Исполнение ЭВМ					
		1	2	3	4	5	6
1	Интерпретирующий диалоговый процессор (ИДП)	+	+	+	+	+	+
2	Накопитель на магнитных дисках (НМД) типа "Изот-1370"	–	+	+	–	–	–
3	Накопитель на гибких магнитных дисках (НГМД) типа "Искра-005-50", "Искра-005-51"	+	+	+	+	–	+
4	Накопитель на магнитных лентах (НМЛ) типа "Искра-005-60", "Искра-005-61"	–	–	+	–	–	–
5	Средства телепроцессинга с СМ и ЕС ЭВМ	+	+	+	+	+	+
6	Графопостроитель "Н-306"	–	–	+	+	–	+
7	Средства сопряжения с СМ ЭВМ	–	–	+	–	–	+
8	Указатель графической информации	–	–	–	–	–	+
9	Средства сопряжения с датчиками	–	–	–	–	–	+
10	Цифроаналоговый преобразователь	–	–	–	–	–	+
11	Аналого-цифровой преобразователь	–	–	–	–	–	+
12	Алфавитно-печатающее устройство (АЦПУ) типа ДЗМ-18	+	+	+	+	+	+

ввод – вывод СПО с НМД, ввод информации с НМД, НГМД, НМЛ, СМ ЭВМ, вывод информации на НМД, НГМД, НМЛ, СМ ЭВМ, графопостроитель (исполнение 3);

ввод – вывод СПО с НМД, ввод информации с НГМД, вывод информации на НГМД, графопостроитель (исполнение 4);

ввод – вывод СПО с НМЛ, обмен информацией с НМЛ (исполнение 5);

ввод – вывод СПО с НМД, ввод информации с НГМД, СМ ЭВМ, аналового устройства, указателя графической информации, цифровых датчиков, НМЛ, вывод информации на НГМД, СМ ЭВМ, аналоговое устройство, графопостроитель, цифровые датчики, НМЛ (исполнение 6).

Элементной базой ЭВМ являются интегральные микросхемы (ИМС) серий К131, К158, К556, К559, К580 и К589, что позволяет надеяться на длительный срок службы ЭВМ "Искра-226". ЭВМ обеспечивает возможность круглосуточной работы пользователя.

Объем оперативной памяти (ОП) ЭВМ "Искра-226" – 128 Кбайт, разрядность – 13 бит. Форма представления чисел при вводе – выводе: целые числа с естественной точкой или экспоненциальная. Диапазон представления порядка чисел при экспоненциальной форме: $-99 < p < +99$. Входной язык – БЕЙСИК версии "WANG – 2200".

Автоматически ЭВМ выполняет следующие операции:

арифметические (+, –, x, :);

сравнения;

нахождения элементарных функций (e^x, возведение в степень, тригонометрические прямые и обратные, $\sqrt{x}$, $\sqrt[x]{}$, ln x, |x|, [x], π, перевод из радианов в градусы и обратно);

нахождения функций с символьными переменными;

команды и операторы языка БЕЙСИК.

Среднее время выполнения операций: арифметических – 0,001 с; вычисления $\sqrt{x}$ – 0,02 с; нахождения элементарных функций – 0,05 с.

Электронные вычислительные машины всех исполнений имеют оперативную управляющую память (ОУП). Дисплей ИДП обеспечивает индикацию символьной и графической информации. Максимальный объем символьной информации – 1920 символов (24 x 80), графической – 256 x 560 точек. ИДП позволяет подключать до семи устройств через интерфейсные блоки (ИБ). Питание ЭВМ осуществляется от сети переменного тока (220В + 10%, 50 Гц).

1.2. Структурная схема ЭВМ

Электронная вычислительная машина "Искра-226" представляет собой вычислительный комплекс, включающий в себя (в зависимости от исполнения) следующие устройства (рис. 1): ИДП, АЦПУ, устройства ввода – вывода УВВ, ИБ для подключения УВВ.

Управляемая ОП (УОП) имеет объем 128 Кбайт и предназначена для хранения программ, данных и микропрограмм (фикси-

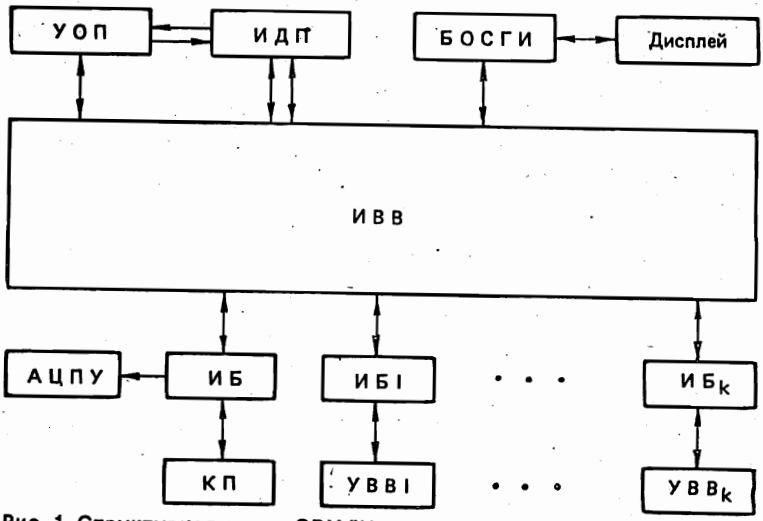

Рис. 1. Структурная схема ЭВМ "Искра-226"

рованных или загружаемых). ИДП служит для управления всем процессом функционирования ЭВМ (обработка информации и управление работой остальных устройств). Интерфейсные блоки $ИБ_1$, ...,$ИБ_k$ ($k = \overline{1,6}$) предназначены для подключения к ИДП устройств ввода – вывода $УВВ_1$, ...,$УВВ_k$ и управления их работой, причем для каждого УВВ (кроме АЦПУ и клавишного пульта КП) имеется свой ИБ, работа которого описывается в технической документации.

Аналого-цифровое печатающее устройство и КП подключаются к ИДП через общий ИБ. АЦПУ содержит буквы латинского и русского алфавитов, арифметические и специальные знаки. Ширина строки – до 158 знаков, количество экземпляров печати – до трех, скорость печати – не менее 40 строк-мин. КП осуществляет обмен информацией по девятиразрядной шине между оператором (пользователем) и ИБ, а через последний – со всей ЭВМ. Работа с КП детально описана в [8].

Блок отображения символьной и графической информации (БОСГИ) предназначен для отображения на дисплее данного типа информации. Можно сказать, что КП и дисплей составляют обычный пульт оператора ЭВМ (консоль).

Интерфейс ввода – вывода ИВВ позволяет организовать многофункциональные системы с магистральной структурой подсоединения $ИБ_1$, ...,$ИБ_k$ к ИДП. Структура и функционирование ИВВ рассматриваются ниже.

1.3. Структура и функционирование управляемой оперативной памяти

Управляемая ОП представляет собой блок полупроводниковой памяти емкостью 128 Кбайт, предназначенной для приема, хранения и выдачи информации. Разрядность слова – 2(1) байта.

Функционально УОП состоит из пяти частей:

1) запоминающей части (ЗЧ);

2) схемы согласования адресной магистрали (СС);

3) схемы приема и выдачи информации (СПВ);

4) схемы управления оперативным запоминающим устройством (СУ);

5) схемы контроля информации (СК).

Запоминающая часть реализована на 36 больших интегральных микросхемах (БИС) и выполняет функции приема, хранения и выдачи оперативной и контрольной информации. СС выдает адреса строк и столбцов запоминающей матрицы в режиме обращения и регенерации. СПВ обеспечивает сопряжение элементов ЗЧ с информационной магистралью ЭВМ в режиме приема и выдачи информации. СУ осуществляет формирование управляющих сигналов RAS, CAS, WE на входах

элементов ЗЧ согласно командам 8/16, МЛ/СТ, РР и А7. СК формирует контрольный разряд каждого байта информации, сравнивает при чтении значения контрольных разрядов (хранящихся в соответствующих элементах памяти) со значениями сложения по mod 2 каждого байта информации, формирует сигнал "СБОЙ КР" при несовпадении сравниваемых значений разрядов.

Управляемая ОП содержит также асинхронный блок управления, реализующий схему приоритета обращения или регенерации. В более широком смысле слова УОП ЭВМ "Искра-226" состоит из следующих частей:

управляющей памяти (УП) емкостью до 64К двухбайтных слов, организованный в виде четырех страниц емкостью 16К х 2 байт;

оперативной памяти емкостью до 32К двухбайтных слов, обеспечивающей запись одного и двух байтов информации (переменный доступ);

памяти микрокоманд (ПМК) емкостью до 2К двухбайтных слов;

сверхоперативной памяти (СВОП) на 30 двухбайтных регистрах общего назначения (РОН).

Функционально РОН делятся на две группы: регистры базовые (РБ) с номерами 0–14 и регистры рабочие (РР) с такими же номерами. Первые используются в качестве базовых при формировании адресов для организации косвенной адресации к памяти и при направленной адресации УВВ, вторые являются рабочими регистрации системы команд ЭВМ и не допускают организации косвенной адресации к памяти и направленной адресации УВВ. Базой сегмента информации в ОП для команд, содержащих непосредственное обращение к ОП, служит РБ13. Указателем верхней ячейки магазина возвратов из программ, который располагается в ОП, является РБ14.

1.4. Структура и функционирование интерпретирующего диалогового процессора

Интерпретирующий диалоговый процессор предназначен для использования в качестве центрального процессора (ЦП) ЭВМ "Искра-226". Он обеспечивает ввод – вывод и обработку информации согласно программе, находящейся в УОП. Ввод программ и интерпретативное выполнение команд и операторов входного языка осуществляются на уровне микропрограмм, фиксированных или загружаемых в УОП.

Интерпретирующий диалоговый процессор реализован на ИМС серий К155, К158, К556, К565 и К589. ИДП исполнения 1 поддерживает УОП емкостью 64 Кбайт, исполнений 2 ÷ 4 – 128 Кбайт. Среднее быстродействие ИДП – порядка 5×10^2

команд/с. Входным языком ИДП исполнения 1 является БЕЙСИК (версия "WANG-2200"), исполнений 2 ÷ 4 – тот же язык, дополненный командами загрузки и редактирования микропрограмм (определяется СПО, фиксированным на уровне команд).

Интерпретирующий диалоговый процессор обеспечивает совмещение во времени обработки и управления вводом – выводом информации. Обмен информацией с УВВ осуществляется в двух режимах: программно-зависимом и в режиме прерываний. Управление УВВ и обмен информацией с ними (включая и устройства, входящие в состав ИДП) выполняются через ИВВ, причем ИДП может одновременно управлять работой нескольких (до семи) УВВ.

Здесь следует заметить, что до сих пор ИДП рассматривался как единая логическая единица и этого было достаточно. Но при рассмотрении системы ввода – вывода на физическом уровне в ИДП необходимо выделить две функционально разные части: ЦП и канальный процессор (КПР). ЦП занимается собственно обработкой и инициацией операций ввода – вывода на логическом уровне, тогда как КПР реализует систему ввода – вывода на физическом уровне. Дальше рассматривается только первая часть ИДП – его ЦП.

Для выполнения системы команд в ЦП имеется ряд служебных регистров: регистр команд (РК) для хранения кода команды на время ее выполнения, РОН, регистр сумматора (РС) в 2 байта, четыре регистра признаков (РП1–РП4), а также двухбайтное арифметико-логическое устройство (АЛУ).

Система команд ЦП содержит следующие основные их типы (в скобках указано количество команд): арифметические и логические (49); направленной адресации устройств ИДП (4); косвенного обращения к ОП (4); табличные (4); передачи управления (6). Следовательно, ЦП работает с 67 основными командами организации обработки данных и управления вычислительным процессом.

Команды по своей структуре двух-одноадресные, т. е. ряд команд обеспечивает выполнение двух операций над двумя операндами. Состояние регистров признаков (РП) определяется после выполнения последней команды.

Арифметические и логические команды обеспечивают выполнение операций бинарной арифметики (+, –) или логических операций (V, сравнение) между содержимым РС и операндами, хранящимися в РОН, или в прямо адресуемых ячейках памяти, а также между константами в самой команде. Например, по команде DD7E к содержимому РС прибавляется содержимое РБ7, а затем – константа 14, и результат засылается в РБ7.

Команды направленной адресации позволяют адресовать канальное устройство (КУ) и ПМК по содержимому РБ. Они

обеспечивают обмен информацией между одним из КУ и РОН или РС, а также переход на микропрограммный уровень управления. Направление адресации задается в 16-м бите РБ. Например, по команде F865 выбирается содержимое РБ6 и анализируется состояние (а) его 16-го бита. Если а = 0, то производится обращение к УП по адресу, сформированному сложением содержимого РБ6, сдвинутого вправо на 1 бит, с содержимым РС. Из УП выбирается однобайтная константа, которая засылается в РР5. Положение константы в слове УП указывает первый бит РБ6: если он равен нулю, то выбирается младшая половина слова, а если единице – старшая. Если а = 1, то производится обращение к КУ, причем служебная информация в КУ поступает из РБ6, а информация из КУ – сначала в РС, а затем в РР5.

Команды косвенного обращения к ОП обеспечивают обмен информацией между ОП и РС или РР, который реализуется по сформированному в заданном РБ адресу. Обмен идет байтами или словами (2 байта). Например, по команде F578 выбирается содержимое РБ7 и производится обращение к ОП по адресу, указанному в битах 16–2 РБ7. Выбранное из ОП слово (2 байта) складывается с содержимым РС и засылается в РР8.

Табличные команды обеспечивают выборку из массива двухбайтной константы и засылку ее в РОН, причем массив констант может располагаться как в произвольном месте УП, так и следовать непосредственно за самой командой. Например, по команде FD45 к содержимому РС прибавляется содержимое РБ4; выбранная по этому адресу из УП двухбайтная константа засылается в РС, а затем в РР5.

Командами передачи управления обеспечиваются: безусловная передача управления в пределах страницы; условная передача; безусловная передача управления в пределах страницы с запоминанием адреса возврата; возврат; переключение по адресу РС; переключение по состоянию РС с запоминанием адреса возврата. Например, по команде A1AC выполняется безусловная передача управления по адресу, указанному в битах 1–14 самой команды, т. е. по адресу 9036 (10).

1.5. Структура и функционирование интерфейса ввода – вывода

Интерфейс ввода – вывода ЭВМ "Искра-226" позволяет организовать многофункциональные системы с магистральной структурой подсоединения ИБ к ИДП. Он содержит 19 шин (рис. 2), предназначенных для побайтного обмена информацией и передачи в ИБ физических адресов устройств (ФАУ).

Шины команд (четыре) служат для передачи в ИБ кодов команд от ИДП, шины состояний (три) – для передачи в ИДП кодов состояний ИБ. По шине вызова ИДП выдает сигнал

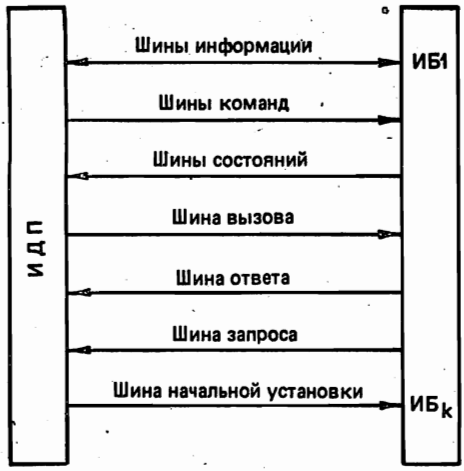

Рис. 2. Структура ИВВ

"Вызов" после получения любой команды из ИБ. По шине ответа ИБ посылает сигнал "Ответ" после окончания приема команды от ИДП. По шине запроса ИБ направляет в ИДП сигнал "Запрос" на обмен информацией через ИВВ. По шине начальной установки ИДП выдает сигнал "Начальная установка", которым все ИБ переводятся в исходное состояние.

Обмен информацией через ИВВ осуществляется побайтно по шине информации с использованием 12 интерфейсных команд, передаваемых ИДП в ИБ, и шести интерфейсных состояний, устанавливаемых ИБ на шинах состояний. Детально функционирование ИВВ описано в [8].

1.6. Система физического ввода – вывода информации

Для реализации операций ввода – вывода информации входной язык БЕЙСИК ЭВМ "Искра-226" содержит 45 операторов, позволяющих пользователю организовать работу с данными на логическом уровне, не прибегая детально к системе ввода – вывода. Однако пользователь может осуществить свой ввод – вывод на физическом уровне посредством обобщенного оператора ввода – вывода ¤GIO, основным назначением которого является обеспечение в рамках языка БЕЙСИК возможности обмена данными между ОП ЭВМ "Искра-226" и соединенными с ней УВВ с помощью программ пользователя.

Оператор ¤GIO позволяет работать и с нестандартными УВВ. Посредством этого оператора запускается программа КПР. Данный оператор содержит ФАУ, непосредственно текст драйвера в шестнадцатиричном коде или указание его места в ОП, а также название буфера для ввода – вывода.

Для программирования обмена информацией с конкретным УВВ на физическом уровне надо точно знать особенности работы этого УВВ и КПР. Рассмотрим подробнее организацию системы ввода – вывода информации ЭВМ.

Электронная вычислительная машина "Искра-226" имеет два процессора: ЦП и КПР, первым из которых выполняются

команды обработки информации и инициации ввода – вывода. По инициации ввода – вывода управление получает КПР, по программе которого выполняются операция ввода – вывода, а также обмен данными между ОП и УВВ.

Канальный процессор связан с УВВ восьмью одинаковыми каналами, по которым одновременно может идти обмен информацией. Каждый канал обеспечен зоной (9 байт), содержащей управляющую информацию для данного УВВ (до некоторой степени зона канала аналогична UCB ЕС ЭВМ в среде операционной системы ЕС ЭВМ [1, 2]):

признаков (прерывания/флага);
состояний (состояние УВВ/номер процедуры обмена);
буфера (данные обмена/константы);
физического адреса УВВ (ФАУ);
адреса текущей команды КПР;
адреса буфера данных в ОП;
длины обмениваемого массива.

Для начала обмена информацией необходимо сформировать зону нужного канала. Делает это ЦП при следующих обязательных условиях:

1) интерпретатор встречает один из операторов ввода – вывода БЕЙСИКа;
2) УВВ, адресованное этим оператором, свободно;
3) данные, необходимые для ввода – вывода, свободны;
4) свободен хотя бы один из восьми каналов КПР.

При выполнении этих условий ЦП формирует зону канала и указывает КП, куда нужно обратиться за драйвером: в ОП или в постоянное запоминающее устройство (ПЗУ). Завершение записи констант в зону канала позволяет КПР начать операцию ввода – вывода данных. Обмен данными между КПР и $ИБ_к$ происходит по ИВВ.

Канальный процессор имеет свою систему команд, составной частью которых являются интерфейсные команды. Однако программирование на физическом уровне требует высокой квалификации пользователя и хорошего знания им особенностей ЭВМ. Поэтому при необходимости разработки драйверов, которых нет в СПО ЭВМ "Искра-226", ее должен вести системный программист [3].

Так как ЭВМ "Искра-226" имеет ярко выраженную магистральную структуру, то ознакомиться в деталях с принципом работы главного интерфейса ЭВМ данного типа можно в [6].

2. АРХИТЕКТУРА И ФУНКЦИИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ЭВМ "ИСКРА-226"

2.1. Состав программного обеспечения

Программное обеспечение ЭВМ "Искра-226" можно разбить на четыре самостоятельные в определенном смысле части (рис. 3): 1) СПО; 2) сервисные средства; 3) обслуживающие средства; 4) ПО пользователя (ПОП). Рассмотрим кратко каждую из них.

2.2. Системное программное обеспечение

Системное ПО, построенное на системе команд ЭВМ, обеспечивает выполнение следующих основных функций:

реализацию входного языка БЕЙСИК;

автоматическое распределение ресурсов ЭВМ между УВВ и программами;

управление процессами ввода – вывода информации;

диагностику работоспособности ИДП.

Функциональные программы, входящие в состав СПО, указаны на рис. 3.

Интерпретирующий диалоговый процессор может использоваться для решения как одной задачи, так и до четырех задач в мультирежиме. Информация о задачах, решаемых в мультирежиме, хранится в ОП процессора. Память между задачами в процессе их постановки распределяет супервизор, на основании сведений о задаче, вводимых пользователем.

В процессе решения задач супервизор обеспечивает защиту областей памяти, занятых задачами, от взаимных наложений. Он также динамически перераспределяет каналы ввода – вывода между УВВ и задачами. Перераспределение может осуществляться и на уровне программно-определяемых прерываний от задач пользователя.

Следующей важной функцией супервизора является регистрация состояния задач в процессе их обработки. Задача, будучи введенной, обрабатывается одним из блоков ИДП, транслятором или интерпретатором. В процессе обработки задача может находиться либо в активном, либо в пассивном состоянии. Задача пассивна, если в этот момент она помещена супервизором в кольцевую очередь.

Квант времени, задаваемый пользователем в ресурсе по задаче, определяет время непрерывного счета, после которого задача прерывается, отправляется в очередь и на ее место из очереди задач выбирается следующая, готовая к выполнению, задача.

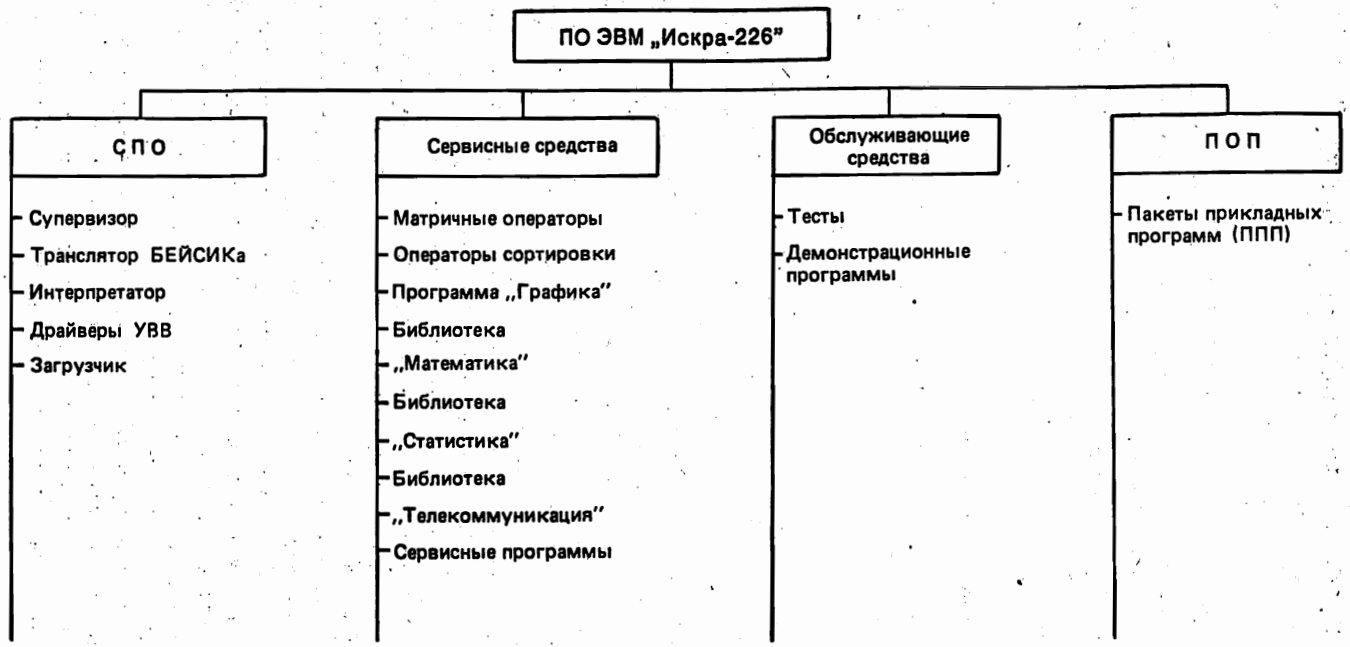

Рис. 3. Состав ПО ЭВМ „Искра-226"

По завершении обработки очередного оператора программы производится проверка таймера по данной задаче с последующим прерыванием ее выполнения, если это требуется. Обслуживание очереди задач выполняется в кольцевом режиме. Задачи располагаются в очереди в порядке загрузки их в ЭВМ. Прерывание процесса обработки задачи и перевод ее в пассивное состояние обусловливается:

ожиданием завершения операции ввода – вывода;

аварийным остановом из-за нехватки ОП, выделенной задаче, или сбоя УВВ;

выполнением операторов БЕЙСИКа, требующих прерывания процесса обработки.

Все сведения о состоянии задач постоянно регистрируются супервизором.

Назначение транслятора – перевод программ с языка БЕЙСИК на машинный язык. Транслятором также решается задача контроля синтаксической и семантической правильности вводимых программ и команд. При любых остановках задача переводится в состояние трансляции, позволяя пользователю редактировать и дополнять программу, причем в процессе редактирования результаты предыдущего счета по возможности сохраняются. Пользователь может после редактирования продолжить счет с использованием ранее полученных результатов.

В процессе ввода программы транслятор обеспечивает полный контроль синтаксиса вводимой строки в ее контексте. Задача анализа синтаксиса и семантики программы в целом решается динамически на этапе счета. В общем случае трансляция с БЕЙСИКа осуществляется в три этапа:

1) трансляция очередной вводимой строки;

2) коррекция программы вычислений после редактирования;

3) формирование зоны значений переменных.

Процесс трансляции очередной вводимой строки программы полностью подчинен синтаксическому управлению, определяемому грамматикой языка БЕЙСИК. В результате очередная строка на машинном языке присоединяется к программе вычислений в ЭВМ и по контексту строки формируются описания переменных.

Основным элементом редактирования является строка программы. Отдельные строки в процессе редактирования могут удаляться, вставляться и добавляться к программе. Поэтому в памяти программа вычислений располагается в порядке поступления строк в ЭВМ. Вычисления же по программе производятся в порядке нумерации строк. Зона значений переменных формируется без просмотра машинной программы на основе построенных описаний переменных.

Интерпретатор, выполняя оператор ввода – вывода в процессе счета по задаче, формирует задание на ввод – вывод и

передает управление супервизору для учета сформированного задания. Супервизор регистрирует полученное задание в очереди заданий на ввод – вывод данной задачи. Задания на ввод – вывод реализуются КПР, который обеспечивает выполнение семи заданий. Остальные задания образуют в каждой задаче очередь на ввод – вывод.

Выбор задания на ввод – вывод из очереди для его выполнения осуществляется по приоритету. Первыми выбираются те задания, которые принадлежат задаче с максимальным приоритетом. Затем проверяется возможность постановки выбранного задания на канал ввода – вывода. Задание может быть поставлено на этот канал, если:

1) свободно УВВ, для которого сформировано задание;

2) имеется свободный канал или данное УВВ подключено к одному из каналов;

3) данное задание удовлетворяет требованиям совместной работы УВВ, закрепленных за каналом.

Если задание не может быть поставлено на канал ввода – вывода для его выполнения, то оно возвращается в очередь заданий. Закончив работу по распределению каналов, супервизор возвращает процессор задаче для продолжения ее выполнения.

Драйверы УВВ осуществляют непосредственное управление устройствами в соответствии со сформированным заданием на ввод – вывод данных.

Язык загрузчика позволяет пользователю вводить, отлаживать, документировать на АЦПУ и внешний носитель микропрограммы СПО. Загрузчик дает возможность провести диагностику ЭВМ. В ходе диалога с загрузчиком сообщения, вводимые в ЭВМ через клавиатуру, пользователь формирует на входном языке загрузчика, пользуясь операторами и командами языка. Среди команд загрузчика следует выделить команду SR, которая может быть введена в ЭВМ "Искра-226" в любом из трех ее состояний: "Загрузчик", "БЕЙСИК" или "Диагностика". При этом выполняются действия по приведению ЭВМ в исходное состояние и устанавливается состояние "Загрузчик".

Подробно работа с загрузчиком описана в [17].

2.3. Сервисные средства

Эти средства построены на основе входного языка БЕЙСИК и реализованы в виде библиотек программ, ориентированных на определенную область применения. БЕЙСИК имеет модульную структуру в том смысле, что каждый его оператор есть основная минимальная логическая конструкция. Основной же

единицей информации в языке является строка, содержащая один или несколько операторов.

Все множество операторов БЕЙСИКа можно разбить на следующие основные группы:

основные операторы;

операторы преобразования информации и функции символьных переменных;

операторы ввода – вывода;

матричные операторы;

операторы сортировки–слияния;

другие операторы.

Для разработки многих программ достаточно первых трех групп операторов (базовый объем) языка, тогда как остальные операторы дают дополнительный сервис пользователю, который он либо использует, либо для своих целей (например, сортировки) пишет собственные программы, используя операторы базового объема.

В языке БЕЙСИК имеется 14 специальных операторов для работы с одно-двухмерными массивами по правилам линейной алгебры. Можно работать как с числовыми, так и символьными данными. Заметим, что наряду со специальными матричными операторами матрицы можно обрабатывать и с помощью некоторых других операторов языка БЕЙСИК. Подробное описание работы с матричными операторами дано в [13].

Для сортировки цифровых и символьных данных язык БЕЙСИК имеет шесть специальных операторов, позволяющих сортировать и (или) объединять массивы информации пользователя [14].

БЕЙСИК располагает 20 специальными операторами для работы с графопостроителем H-306, что позволяет пользователю на высоком уровне программирования разрабатывать программы типа "Графика". Подробная инструкция по использованию этого сервисного средства имеется в [16].

Библиотека "Математика" содержит 45 программ для решения задач разной степени сложности из различных разделов математики: извлечение корней, интегрирование и дифференцирование, линейная алгебра, тригонометрия, геометрия, теория чисел, комплексные числа, специальные функции и гармонический анализ. Программы составлены для наиболее типичных задач с использованием алгоритмов, разработанных фирмой "WANG Laboratories" в 1973 г. для ЭВМ "WANG-2200". Подробные инструкции по работе с этими программами помещены в [21].

Библиотека "Статистика" содержит 40 программ на языке БЕЙСИК для решения задач разной степени сложности из различных областей статистики и техники: регрессия, дисперсионный анализ, теория вероятностей и др. Алгоритмы про-

грамм разработаны фирмой "WANG Laborаties" в 1973 г. для ЭВМ "WANG – 2200". Подробная инструкция по работе с этими программами имеется в [22].

Библиотека "Телекоммуникация" содержит прикладные программы для обеспечения возможностей пользователю использовать стандартные технические и программные средства телеобработки СМ и ЕС ЭВМ. Подробное описание этих программ дано в [23].

Большое количество сервисных программ предназначено в первую очередь для облегчения работы пользователя с наборами данных на различных накопителях (НГМД, НМД, НМЛ), подготовки накопителей к работе, организации локальных баз данных и данных в нужном для пользователя виде и т. д. Организованы они, как правило, в виде ППП. Заметим, что многие из этих средств зависимы от версий входного языка БЕЙСИК, что должно учитываться при их эксплуатации.

Среди таких средств отметим несколько наиболее интересных:

СОПИ – ведение массивов форм и числовых данных; печать форм унифицированных документов; фиксированные, псевдофиксированные и переменные формы; связь между показателями различных форм [24];

СПОТ – ведение таблиц; редактирование текста и его запись на НМЛ ЕС ЭВМ [25];

НИВА – ввод и запись на НМД или НГМД данных; считывание и печать таблиц, формирование и счет в диалоге; свод таблиц в файлы; выход на базу данных (БД) "Колос"; инициализация НМД [26];

АСОД (адаптируемая система обработки данных) – подготовка тома; размещение библиотеки и работа с ней; получение справок с НМД; работа с НМЛ;

PRINT – печать в требуемой форме выходных файлов;

PZP – перезапись файлов с диска на диск полностью или выборочно;

MOVE – копирование файлов с диска на диск с переименованием;

ВЕЛОБАД – программно-технологический комплекс ведения баз данных [45];

ПУДИС – пакет программ управления данными в информационных системах с элементами реляционных баз данных [32];

АССЕМБЛЕР – пакет программ для автоматизации разработки программ на языке инструкций ЭВМ "Искра-226" [35].

В настоящее время имеется достаточно большой набор пакетов программ как общего, так и специального назначения, позволяющих пользователям из различных сфер деятельности

успешно использовать ЭВМ "Искра-226" как по ее прямому назначению, так и в качестве интеллектуального терминала в вычислительных сетях обработки информации [29].

2.4. Обслуживающие средства

В состав этих средств входят тесты и демонстрационные программы. Комплект контрольных тест-программ (ККТП) поставляется на магнитных носителях (НГМД, НМЛ), как правило, отдельно от СПО и служит для всесторонней проверки ЭВМ "Искра-226", т. е. функционирования ее ИДП (12 тестов), УВВ (9 тестов), функционирования различных исполнений ЭВМ (6 тестов) и функционирования ЭВМ в автоматическом режиме (2 теста, включая тест проверки производительности ЭВМ).

Работа с тестами довольно проста. После включения ЭВМ на дисплее появляется сообщение "ЗАГРУЗЧИК", после чего в ЭВМ загружается остальное СПО на НГМД или НМЛ. Для исполнения 3 ЭВМ, которое дальше используется в примерах, загрузка идет по директиве

LOAD $\{\begin{matrix}F\\R\end{matrix}\}$#K,

где K – номер загружаемой копии СПО; F, R – нижний и верхний дисководы.

После успешной загрузки на дисплее появится сообщение
BASIC 01 ⟨дата⟩,
и теперь надо выдать директиву RUN 1.

После сообщения READY ЭВМ готова к работе. Поскольку ККТП находится отдельно, для ввода нужного текста выдается директива

LOAD DC $\{\begin{matrix}F\\R\end{matrix}\}$ ⟨имя теста⟩

Перед началом работы с ККТП рекомендуется директивой LIST DC $\{\begin{matrix}R\\F\end{matrix}\}$ вывести на дисплей перечень имеющихся в библиотеке тестов. После появления на дисплее сообщения ":–" следует выдать директиву RUN и работать с тестом согласно его описанию. Результаты работы теста выводятся на дисплей и АЦПУ. Перед вводом очередного теста необходимо вводить директиву CLEAR, по которой содержимое памяти восстанавливается. Подробные сведения о работе с ККТП имеются в [19].

Заметим, что наряду с ККТП средствами тестирования располагают и средства загрузчика, содержащие восемь диагностических тестов, которые позволяют диагностировать основные устройства ЭВМ. Но в отличие от ККТП работой этих

тестов управляет сам загрузчик, который после получения на дисплее сообщения "ЗАГРУЗЧИК" выдает директиву RUN T и по ней начинает выполняться группа тестов диагностики процессора.

По мере их прохождения на дисплее отображается название очередного теста и, если он выполнен правильно, слово "ПРОШЕЛ". Затем проверяется контрольная сумма ПЗУ загрузчика. Если она правильная, то на дисплее отображаются конфигурация ЭВМ и перечень ФАУ, входящих в состав конфигурации ЭВМ:

ФАУ	УВВ
01	Клавиатура
0C	АЦПУ (ДЗМ-18)
14	Графопостроитель Н-306
18	НГМД
1C	НМД
08	НМЛ
05	Дисплей

Далее на дисплее отображается сообщение ":–" и управление получает пользователь. Если контрольная сумма ПЗУ загрузчика неправильная, то на дисплее появится сообщение "СБОЙ ПО КС ПЗУ", а после нажатия клавиши CR/LF – конфигурация ЭВМ, и можно выполнять тесты. В том случае, если один из тестов не прошел процессора, на дисплее у этого теста не появится слово "ПРОШЕЛ", и ЭВМ автоматически запускает следующий тест.

После вывода на дисплей конфигурации ЭВМ следует нажать клавишу CR/LF, в результате чего на дисплее появится перечень тестов. Задав директиву номер теста CR/LF, получим выполнение соответствующиего теста. Для возврата ЭВМ в состояние "Загрузчик" необходимо нажать кнопку SR (системный сброс) или ввести директиву 8 CR/LF. Подробные сведения о работе с тестами загрузчика имеются в [17].

К обслуживающим средствам можно отнести демонстрационные задачи и игры, которых насчитывается уже немалое количество. В комплектность ЭВМ "Искра-226" входит набор трех таких задач и пяти игр. Принцип работы с ними такой же, как и с ККТП, и рассмотрен в [20].

Демонстрационные задачи дают пользователю рекламные данные об ЭВМ и ее программных возможностях. Игры позволяют ему в игровой форме легче приобщиться к работе с ЭВМ в диалоговом режиме и в определенной мере организовать интеллектуальный отдых. Но не только в этом польза подобных задач. Составленные с использованием многих средств языка БЕЙСИК и содержащие, порой, немного операторов, выведен-

ные листинги их программ на АЦПУ могут служить хорошими образцами для пользователя, начинающего заниматься программированием на языке БЕЙСИК.

2.5. Программное обеспечение пользователя

Программное обеспечение, разработанное пользователями для различных областей его применения, составляет подавляющую часть всего ПО ЭВМ "Искра-226", если учесть также весь тот объем ПО (совместимого с ПО ЭВМ "Искра-226"), который был собран более чем за 10 лет фирмой "WANG Laboratories" и другими фирмами для ЭВМ " WANG-2200".

Поэтому перед пользователем часто стоит нелегкая задача из всего обилия имеющегося ПО выбрать для себя наиболее приемлемое на тот или иной случай. К сожалению, это не единственная задача. Ниже рассматриваются более конкретные вопросы использования ЭВМ "Искра-226".

2.6. Параллельное использование ресурсов ЭВМ

Как уже отмечалось, ЭВМ "Искра-226" имеет два процессора (ЦП и КПР), которые работают параллельно на общую для них ОП.

Оперативная память служит для хранения программ, данных и результатов их обработки. Она управляется ЦП или КПР, причем ЦП содержит встроенные программы трансляции и интерпретации, а также встроенную операционную систему, обеспечивая подготовку заданий КПР и запуск его для ввода – вывода информации. Задание на ввод – вывод данных представляет собой совокупность процедур ввода – вывода. КПР в соответствии с заданиями управляет УВВ и осуществляет обмен информацией между ОП и нужными УВВ.

Таким образом, отличительной особенностью ИДП является возможность параллельного выполнения операций обработки и обслуживания УВВ (причем одновременно обслуживается до семи УВВ) за счет независимой работы ЦП и КПР.

Реализация процедур ввода – вывода информации осуществляется в мультиплексном режиме по нескольким каналам (каждой процедуре выделяется свой канал). Максимальное число каналов, параллельно обслуживаемых КПР, – 8 (с нулевого по седьмой, причем канал 7 всегда занят таймером). Дисциплина обслуживания каналов – кольцевая, т. е. после выполнения одной команды КПР по данному каналу осуществляется активация следующего и т. д. (по кольцу). По окончании процедуры КПР прерывает работу ЦП для получения новой процедуры ввода – вывода информации.

В каждый момент времени к магистрали ввода – вывода данных может быть функционально подключено только одно УВВ. Включение ЭВМ или нажатие кнопки " SR " устанавливает систему в исходное состояние, и вся ОП со всеми подключенными УВВ предоставляется одной задаче. Затем на консоль вывода оператора выводится сообщение о готовности ЭВМ к работе, а управление передается пользователю. Исходное состояние ЭВМ можно изменять посредством оператора SELECT.

3. РАБОТА С МАГНИТНЫМИ НАКОПИТЕЛЯМИ

В качестве основных магнитных накопителей в ЭВМ "Искра-226" используются НМД и НГМД. Первые состоят из съемного и фиксированного жестких дисков, вторые содержат два гибких диска (флопи-диски). Каждый пакет для НМД (верхний съемный R, нижний – фиксированный F) имеет 408 треков. Информационная емкость жесткого диска – 9792 сектора по 256 байтов в каждом, а одной поверхности гибкого диска – – 1001 сектор и тоже по 256 байтов в секторе. Каждая поверхность гибкого диска имеет 77 треков, в каждом треке по 13 секторов (нулевой трек – индексный, 76-й трек не используется).

Работа НМД возможна только при наличии обоих дисков (съемного и фиксированного). НМД и НГМД имеют аппаратные средства защиты от записи. Подготовка дисков к работе подробно описана в [12].

Для обращения к конкретному диску необходимо его адресовать. Тип диска задается параметром: R – съемный, F – фиксированный, T – любой (R или F) диски. Само устройство задается его адресом ФАУ : IC – НМД; 18 – НГМД.

БЕЙСИК позволяет работать с дисками обоих типов в двух режимах: режиме каталога файлов и режиме адресации секторов. В первом режиме ЭВМ автоматически следит за размерами и размещением каждого каталогизированного файла (программы или данные), что существенно упрощает работу пользователя с дисками, так как многие сложные функции по обслуживанию файлов берет на себя сама система. Режим адресации секторов дает возможность пользователю работать с любыми секторами на диске путем задания адреса сектора, но пользователь в этом случае должен сам следить за объемом и размещением информации на диске, что часто требует разработки ряда дополнительных программ и особой внимательности при работе с файлами.

3.1. Инициализация дисков

Перед использованием диск любого типа необходимо проинициализировать. Эта операция состоит в разметке поверхности и занесении служебной информации. Для конкретного носителя данная операция относительно редка, но ее рекомендуется проводить для каждого накопителя после его длительного использования.

Сектор диска содержит 260 байт, из которых 2 байта отведено на адрес сектора, 2 байта – на контрольную информацию, остальные 256 байт – на информацию пользователя. При инициализации заполняются первые 4 байта каждого сектора, в остальные байты секторов заносятся шестнадцатиричные нули. Для инициализации можно использовать следующую программу:

```
 10 SELECT PRINT XX                                    R
{15 PRINT   HEX (1B000100000400)}   для диска
{15 PRINT   HEX (1B000000000400)}     "-"             F
 20 END
```

где XX = 18 для НГМД и XX = 1C для НМД.

Вопросы рационального использования поверхности диска при записи на него данных рассмотрены в [12].

3.2. Режим каталога файлов

В этом режиме используется 17 операторов, осуществляющих управление файлами. Эти операторы позволяют создавать файлы информации (программы и данные) на дисках, а также обращаться к ним по имени файла (а не по адресам секторов). Каждый вновь создаваемый файл помещается в свободную область диска; его имя, адрес и другая служебная информация о нем помещаются в указатель каталога. Помимо этого, имеются другие интересные возможности работы с файлами: переход от записи к записи в самом файле, копирование файлов и выдача основных характеристик файла. Каталог состоит из указателя каталога (УК) и тела каталога. Запись каталогизированных файлов производится на диске последовательно сектор за сектором.

Обращение к каталогизированному файлу осуществляется следующим образом: в УК отыскивается имя файла и по нему выбирается начальный адрес файла на диске. Признаком режима каталога файлов является указатель DC, который следует за названием оператора. Например, по команде

```
10 LOAD DC R  "СКБ"
```

в память загружается программа с именем "СКБ", расположенная на верхнем НМД.

Перед работой в режиме каталога на диске необходимо создать каталог с помощью оператора SCRATCH DISK. При записи УК следует помнить, что в нулевом секторе можно поместить информацию о 15 файлах, в остальных – о 16 файлах (имя файла занимает 8 байтов). Оператор SCRATCH DISK предназначен для резервирования секторов под УК и самим телом каталога. Указатель LS оператора задает объем УК ($\leqslant$ 255 байт).

Тело каталога следует сразу за УК, и его граница задается указателем END оператора. Например, по команде

10 SCRATCH DISK R #5, END = 1285

отводятся на верхнем НМД с логическим номером 5 под УК 24 сектора (0 – 23), а под его тело – секторы 24 – 1285. По команде же

15 . SCRATCH DISK R #5, LS = 10, END = 1285

в отличие от первого примера под УК отводится 10 секторов (0–9).

Целесообразно оставлять место на НМД за телом каталога для возможности организации работы с временными рабочими файлами. Тело каталога можно уменьшать или увеличивать с помощью оператора MOVE END,, тогда как изменение объема УК требует полной реорганизации всего каталога. Это следует учитывать при первоначальной его организации. После подготовки каталога операторы режима каталога файлов можно использовать для записи, чтения и управления файлами данных и программ.

Для рассматриваемого режима работы с файлами надо уметь работать с таблицей УВВ (ТУВВ), которая представлена в памяти в виде восьми строк, имеющих каждая свой номер файла. Если дисковым оператором не задан номер файла, то обращение идет к консольному дисковому УВВ, т.е. УВВ, занесенному в нулевую строку ТУВВ.

Строка ТУВВ содержит:

номер файла (тождественно равен номеру строки и логическому номеру УВВ);

физический адрес УВВ, содержащего данный файл, и тип диска;

адреса начального и последнего секторов файла.

Физический адрес устройства записывается в ТУВВ по команде

SELECT ⟨логический номер⟩⟨ФАУ⟩.

Например, по команде 50 SELECT # 418, # 618 дисковому УВВ с ФАУ 18 присваиваются логические номера 4, 6 и отводятся четвертая и шестая строки ТУВВ. Теперь по команде 70 LOAD DC R #4 "МПСМ" в память будет загружаться программа с именем "МПСМ" с верхнего УВВ, описанного в четвертой строке ТУВВ. В операторах обработки файлов данных в режиме

адресации секторов разрешается лишь косвенное задание адреса дискового УВВ – через номер файла.

Для доступа к файлу необходимы следующие данные о нем:

тип НМД и адрес дискового УВВ с файлом;

адреса начального, текущего и конечного секторов файла.

Эти данные переписываются из УК на диске в ТУВВ каждый раз, когда открывается файл данных (впервые или повторно). Адрес текущего сектора полагается равным адресу начального сектора файла. Файл открыт, если информация о нем помещена в ТУВВ, и закрыт, если эта информация удалена из ТУВВ. Например, по команде

```
30 SELECT #5  18
40 DATA SAVE DC OPEN R     #5 (42) "СИЛИКААТ"
```

открывается файл "СИЛИКААТ" на верхнем НМД с ФАУ 18, чьи логические записи будут занимать 42 сектора диска.

Подробно работа с файлами на дисках в режиме каталога файлов описана в [12] и будет рассмотрена ниже.

3.3. Режим адресации секторов

Восемь операторов режима дают возможность читать информацию, записывать и копировать ее на другой диск, проверять качество записи. В этом режиме не обеспечивается автоматическое создание и ведение УК. Забота о хранении и обработке всей адресной информации ложится на самого пользователя.

В данном режиме выделяют два типа операторов: DA и BA. Операторы первого типа используют для записи и чтения программ и данных, начиная с указанного в них номера сектора. Если информация занимает более одного сектора, то запись в нужное число секторов идет автоматически. Форматы представления данных для режима каталога файлов и режима адресации секторов совпадают, что позволяет читать с помощью операторов типа DA информацию, записанную в режиме каталога файлов.

Операторы типа BA специально ориентированы на работу с информацией только одного сектора. Читать с диска посредством оператора типа BA можно информацию, записанную с помощью любого дискового оператора.

Таблица УВВ для режима адресации секторов существует и содержит следующую информацию:

начальный адрес сектора, заданный в операторе указателем ⟨адрес сектора⟩ ;

максимально возможный номер сектора (определяется типом диска);

номер сектора, следующего за последним использованным

сектором по данному оператору; после реализации оператора это значение получает указатель ⟨переменная⟩, заданный в операторе.

Выбор строки ТУВВ аналогичен ее выбору в режиме каталога файлов. При чередующемся использовании операторов обоих режимов необходимо быть особенно внимательным. Прямая адресация позволяет также копировать и проверять хранимые на диске данные (операторы COPY и VERIFY), а также программно обрабатывать статус программы и данных: признаки защиты и удаляемости файлов.

3.4. Временные рабочие файлы на дисках

Временные рабочие файлы открываются оператором режима каталога файлов, но данные о них не заносятся в УК и не записываются в тело каталога. Сами параметры временных рабочих файлов заносятся в ТУВВ. Для доступа к ним используются операторы режима каталога файлов, но вместо имени файла применяется специальный указатель TEMP, задаваемый в операторе открытия файла при первом к нему обращении. Можно работать с несколькими временными рабочими файлами. Подробную информацию о работе с дисковыми файлами всех режимов можно найти в [12].

Наряду с НГМД и НМД в ЭВМ "Искра-226" используются и магнитные ленты, но ввиду относительно редкого их применения в устройствах прямого доступа, а также значительно меньшей надежности при эксплуатации, здесь не будем рассматривать работу с ними, а заинтересованного читателя отсылаем к [11].

4. ОСНОВЫ ПРОГРАММИРОВАНИЯ НА ЯЗЫКЕ БЕЙСИК

4.1. Общие сведения

БЕЙСИК является входным языком ЭВМ "Искра-226" и полностью соответствует входному языку "WANG-2200". БЕЙСИК – язык высокого уровня, предназначенный для программирования различных задач и отличающийся простотой программирования и обращения к периферийному оборудованию, доступностью и легкостью изучения. Он позволяет пользователю работать с цифровой, символьной и графической информацией.

БЕЙСИК имеет строчную структуру, т. е. основной единицей информации является строка, которая может быть программной или строкой непосредственного счета (НС). Программная строка имеет номер и один или несколько операторов, строка

НС не имеет номера и состоит также из одного или нескольких операторов.

Строки обоих типов строятся из операторов. Оператор – основная минимальная логически законченная конструкция языка, содержащая имя оператора и его параметры. Оператор определяет объекты, тип и последовательность действий над ними. Операторы в строках разделяются двоеточием (:). Строка при вводе в ЭВМ не обрабатывается до нажатия клавиши CR/LF, после чего:

при вводе строки НС осуществляется синтактический контроль и при отсутствии ошибок строка выполняется;

при вводе программной строки выполняется синтактический контроль, и текст заносится в память.

Алфавит БЕЙСИКа включает в себя следующие подмножества символов:

латинский алфавит (26 букв);

русский алфавит (31 буква);

десятичные цифры (0–9);

шестнадцатиричные цифры (0 – F);

специальные знаки (!, ", # , ¤, %, & и т. д.).

Основными объектами обработки являются:

цифровые и символьные константы и переменные (включая элементы массивов);

одно- и двухмерные цифровые и символьные массивы;

массивы графических изображений.

Если значение объекта в программе не задано, то цифровой объект имеет нулевое значение, символьный – значение пробела. Все операции, допустимые в языке, выполняются над значениями соответствующих объектов. Для работы в среде БЕЙСИКа пользователь должен выполнить следующие основные действия:

1) включить ЦП и все необходимые УВВ (на дисплее при этом появится сообщение "ЗАГРУЗЧИК");

2) поставить на выбранное дисковое УВВ накопитель с версией языка БЕЙСИК, привести УВВ в готовность;

3) ввести директиву LOAD {R/F} # K (К – номер копии);

4) после получения сообщения " BASIC 01 ⟨дата⟩ " ввести директиву RUN 1 (при этом ЦП переходит в состояние ожидания с сообщением " READY " и готов к приему директив и программ пользователя).

4.2. Простейшие конструкции языка и его возможности

В ЭВМ используются цифровые переменные, позволяющие приписывать числовые значения некоторому идентификатору. Этот идентификатор состоит из буквенно-цифровой комбинации, где первой должна быть буква. В среде БЕЙСИКа мож-

но использовать 286 разных обозначений цифровых переменных, в задаче же – одновременно не более 208 различных индентификаторов. Признаком целой переменной является символ %. Например, X% = 5, X = 5 (в первом случае X получает целое значение 5, во втором – действительное 5).

В ряде операций допускается совместное использование обоих типов переменных, например B = C + C%. Значение действительной переменной лежит в интервале $10^{-99} \leqslant |A| \leqslant 10^{99}$, целой – в интервале $0 \leqslant A\% \leqslant 7999$. Операции с целыми переменными более быстрые (и существенно). Примеры:

C= SQR (A^5 + B^7), т. е. $C = \sqrt{A^5 + B^7}$;

X3= 4/3*5^3, т. е. $X = (4/3) * 5^2$.

Индикация значения переменной на дисплее или печати осуществляется по оператору PRINT. Например, по PRINT C,X3 и PRINT /OC, C,X3 печатаются значения переменных C и X3 соответственно на дисплее и АЦПУ.

БЕЙСИК обеспечивает выполнение основных арифметических операций: сложения (+), деления (/), вычитания (–), умножения (*) и возведения в степень (^). Для изменения естественного порядка их выполнения используются скобочные записи выражений. ЭВМ позволяет вычислять без программирования 14 часто используемых математических функций: тригонометрических, натурального логарифма, e^x, генерации псевдослучайных чисел и т. д. [10].

Для ввода чисел в ЭВМ служит группа алфавитно-цифровых клавиш. Количество цифр при вводе числа – не более 13. Числа вводятся в двух формах: естественной и экспоненциальной. При вводе чисел в экспоненциальной форме сначала вводится мантисса числа, затем – признак порядка (E), знак порядка и, наконец, порядок вводимого числа. Например, число 6.4E + 5.

Наряду с цифровыми данными БЕЙСИК позволяет работать с символьными: константой, переменной и массивом. Идентификатор простой символьной переменной содержит в конце знак ¤, например, A1 ¤, B2 ¤ и т. д. Стандартная длина символьной переменной должна быть $\leqslant 16$; длина ее объявляется операторами DIM и COM в пределах 1–253. Наличие HEX – функции в языке дает возможность пользователю выводить любые шестнадцатиричные коды на любое УВВ. Примеры:

B¤= "CKB MPSM", M¤= "468cil"

Следует помнить, что символьная информация в ЭВМ на языке БЕЙСИК вводится одним из четырех способов:

1) без ограничителей; 2) в кавычках ("); 3) в апострофах; 4) посредством HEX – функции.

С другой стороны, буквенная информация (русский или латинский алфавит) может вводиться в двух регистрах: верхнем (прописные буквы), нижнем (строчные). Ввод буквенной информации первым и вторым способами соответствует верхнему регистру, а третьим способом – нижнему регистру. Например:

```
                    HEX (41)
ABZagnA
```

Информация по HEX–функции вводится согласно шестнадцатиричному представлению символов.

HEX – функция в языке дает возможность выборки любых символов в символьной переменной; она может находиться по обе стороны равенства. Например, в результате обработки выражения STR(B¤,4,4)=STR(M¤,1,4) символьная переменная B¤ получит значение "СКБ4684М", а символьная переменная M¤ не изменится.

Символьные операции не используются в арифметических выражениях, но с ними можно выполнять операции сравнения. Подробно символьные переменные рассматриваются в [10, 11, 18].

Наряду с простыми цифровыми и символьными переменными БЕЙСИК позволяет работать с одно- и двухмерными цифоровыми или символьными массивами, причем массив идентифицируется именем, размерностью и объемом, как и в других языках.

Например, массив A(1), B¤ (K), C (1,K), M¤ (1,K);
его элементы A (1), B¤ (8), C (3,4), M¤ (5,6).

Оператор COM M (6) объявляет одномерный массив длиной 6. Индексация в массивах идет от 1 до максимального значения индекса координаты.

В строке НС может быть более одного оператора; при этом операторы разделяются символом (:). Например: PRINT 45: PRINT B (4). Выполнение строки НС реализуется сразу же после нажатия клавиши CR/LF. В этой строке можно использовать большинство операторов БЕЙСИКа, составлять небольшую программку и сразу же выполнять ее.

Строки программ на языке БЕЙСИК до нажатия клавиши CR/LF очень просто корректируется клавишами BACKSPACE и LINEERASE, а после ее нажатия (кроме строк НС) их можно корректировать в режиме EDIT клавишами DELETE, ERASE, INSERT, BACKSPASE, LINEERASE и шестью клавишами, управляющими движением курсора [11].

Откорректированная строка выполняется после нажатия клавиши CR/LF.

4.3. Порядок подготовки и выполнения программ

После детального уяснения алгоритма задачи пользователь, как правило, составляет его схему с достаточной для целей программирования детализацией. Затем, используя операторы языка БЕЙСИК, программист отображает схему алгоритма в виде программы, которая состоит из строк, пронумерованных от 0 до 9999. Номер строки идентифицирует ее, позволяя обращаться к строке по ее номеру. Как правило, нумерацию делают через некоторый шаг, чтобы впоследствии можно было вставлять нужные строки.

После составления программы ее вводят с дисплея в ЭВМ построчно, осуществляя при необходимости нужную корректировку вводимого текста с использованием описанных выше средств редактирования. Перед вводом программы рекомендуется очистить память посредством оператора CLEAR. Номер строки можно вводить либо непосредственно, либо с помощью клавиши STMT NUMBER с шагом 10, а сами строки – в произвольной последовательности их номеров.

После ввода всей программы пользователь может вызвать ее полный текст на просмотр посредством оператора LIST. Программа выводится страницами (по 23 строки) на дисплей или печать в порядке возрастания номеров строк. Режим вывода программы определяется данными оператора LIST. Например, с помощью операторов LIST S и LIST /OC выводится соответственно на дисплей весь текст программы постранично и на АЦПУ.

При корректировке строки, находящейся в памяти ЭВМ, достаточно ввести откорректированный текст строки с тем же номером и нажать клавишу CR/LF. Если участок программы не имеет резерва номеров, то с помощью оператора RENUMBER можно перенумеровать всю или часть программы.

Полностью скорректированный текст программы находится в памяти ЭВМ и для ее трансляции, редактирования и выполнения достаточно нажать клавишу RUN. Программа выполняется в естественном порядке ее строк, если не предусмотрен иной порядок (переходы, циклы и т. д.). Для повторного считывания нужно еще раз нажать клавишу RUN. В программе может использоваться оператор STOP, осуществляющий ее останов; считывание возобновляется после нажатия клавиш CONTINUE или HALT/STEP. Число операторов STOP в программе не ограничивается. Помещение в конце программы оператора (необязательного) END позволяет получать на дисплее в конце программы информацию об оставшейся свободной памяти ЭВМ.

Очень важным моментом (если не главным) является отладка составленной программы. В настоящее время име-

ется довольно много способов эффективной отладки как системного, так и проблемного ПО. На этом вопросе останавливаться не будем, а заинтересованного читателя отсылаем к хорошо известным книгам по ПО. Только об одном моменте, связанном со спецификой возможностей БЕЙСИКа, следует сказать особо. Использование клавиши HALT/STEP позволяет исполнять программу пооператорно.

Каждое нажатие указанной клавиши ведет к выполнению одного оператора. Для эффективного пользования этим средством надо использовать его совместно с оператором STOP, который помещается в самом начале программы. Например:

```
10 STOP
20 A = 545: B = 14
.....
110 PRINT A, B
120 END
```

При нажатии клавиши RUN программа, начав работать, сразу же выходит на останов, который снимается нажатием клавиши CONTINUE или HALT/STEP, что позволяет выполнять программу пооператорно. Если пооператорный режим работы нужно прекратить, то надо нажать клавишу CONTINUE. С помощью оператора TRACE пользователь может получать значительный объем отладочной информации [11].

После целого ряда возникающих сбоев (включая и некоторые случаи SYSTEM 00) при работе с программой иногда можно продолжить ее выполнение без перезагрузки программы. Нажатием клавиши RESET программа остается в памяти ЭВМ (что следует проверить директивой LIST CR/LF) и работу можно продолжить по RUN ⟨номер строки⟩.

Подробные сведения о возможностях отладки программ на языке БЕЙСИК можно найти в [11]. Отметим только, что отладка программ на ЭВМ "Искра-226" намного проще аналогичной процедуры на ЕС ЭВМ и СМ ЭВМ [1–6, 28].

4.4. Основные операторы

Ниже приводится краткое описание основных операторов языка БЕЙСИК. Подробное их описание с учетом всех возможностей дано в [10, 11, 18]. Здесь же приводятся самые необходимые сведения и рассматриваются некоторые особенности выполнения операторов.

Оператор CLEAR позволяет очистить память ЭВМ от предыдущих данных и программ. С его помощью (без дополнительных данных) стирается в памяти вся программа, тогда как по CLEAR V стираются значения всех переменных,

а текст программы сохраняется. Рекомендуется этим оператором пользоваться всякий раз перед очередным вызовом программы на отладку или выполнение. В противном случае результат, вообще говоря, будет непредсказуем.

Оператор COM служит для объявления переменных и массивов, общих для нескольких программ, а также для резервирования места в памяти ЭВМ для массивов и задания длины символьных переменных и (или) элементов символьного массива. COM должен выполняться до использования любых переменных и массивов, перед первым оператором DIM . Переменные и массивы, определенные по COM, являются общими для нескольких программ, и их значения не изменяются при вводе этих программ в память. При наличии общих переменных, используемых в нескольких последовательно выполняемых программах, оператор COM может быть только в первой программе.

Значения общих переменных стираются лишь по CLEAR, CLEAR V. Оператор COM CLEAR предназначен для переопределения всех или части общих переменных (массивов) в необщие и наоборот.

Оператор DATA не участвует в непосредственном считывании программы, а служит для хранения констант, значения которых могут присваиваться переменным оператором READ. При наличии нескольких операторов DATA они реализуются последовательно. Для многократного использования данных, определенных оператором DATA, следует использовать оператор RESTORE. Например:

```
20 DATA 42.47, – 42.67, "СКВ", 2, "ABC"
30 READ X,Y,Z, D %, C¤
40 PRINT X,Y,Z, D %, C¤
```

Оператор READ служит для присвоения констант переменным, заданных оператором DATA. В строке НС оператор READ имеет смысл только после оператора RUN.

Оператор DEFFN предназначен для определения функций пользователя. Обращение к такой функции можно сделать в любом месте программы с помощью функции FN. Например:

```
10 DEFFN A(X)=3*X^2+5
20 X j 1:Y= FN A(X)
30 PRINT X,Y
```

Оператор DEFFN' служит для ввода в программу часто встречающихся текстов посредством 16 клавиш СФ, а также для организации помеченных программ. Пусть, например, выражение "ТЕОРИЯ ОДНОРОДНЫХ СТРУКТУР" часто встречается в программе; будем вводить его клавишей 5 СФ:

10 DEFFN' 5 "ТЕОРИЯ ОДНОРОДНЫХ СТРУКТУР". Теперь, чтобы ввести строку 20 PRINT "ТЕОРИЯ ОДНОРОДНЫХ

СТРУКТУР", достаточно нажать клавиши 20 PRINT"5 (зона СФ)" CR/LF .

С другим назначением оператора DEFFN' можно ознакомиться в [11]. Эта его форма используется для организации помеченных подпрограмм, являющихся сильным инструментальным средством языка БЕЙСИК при разработке программ сложной структуры и программ-интерпретаторов. Обращение к помеченным подпрограммам осуществляется посредством оператора COSUB' с формальными данными или без них.

Оператор DIM предназначен для резервирования памяти под массивы, а также для задания длины символьных переменных. Он должен помещаться в программе перед первым обращением к этим массивам и переменным. Например: 10 DIM A(50); 20 DIM A ¤ 40; 50 DIM M(47), P(67), B(5,5); 340 DIM K ¤ (256)1, H% (128), X(128,128).

Оператор END определяет конец программы и является необязательным. После его выполнения программа завершается, и на дисплее высвечивается сообщение:

END PROGRAM

FREE SPACE количество байтов свободной памяти

Использование этого оператора в строке НС позволяет в любой момент идентифицировать свободную память.

Оператор ¤GIO' предоставляет возможность написания фрагментов программ в машинных кодах, точнее, на автокоде TEMPO [8].

Оператор REM позволяет включать комментарий в текст программы. Этот оператор не выполняется. Текст после оператора REM можно вводить без кавычек, но в тексте не должно быть двоеточий, поскольку они служат разделителями операторов в строке. Например:

20 REM ОРГАНИЗАЦИЯ ЦИКЛА
90 REM "ПРОГРАММА СОЗДАНИЯ КАТАЛОГА ПРЕДПРИЯТИЙ МПСМ"

Оператор RENUMBER служит для перенумерации строк программы, а также ссылок на них. Он содержит от одного до трех позиционных данных (параметров). Первый параметр указывает номер строки, которая подлежит перенумерации; при его отсутствии будет перенумерована вся программа. Второй параметр обозначает число, которое становится новым номером строки, с которой начинается новая нумерация; если он отсутствует, то полагается равным значению третьего параметра, который определяет шаг новой нумерации (если третий параметр не задан, то принимается значение 10). Следовательно, посредством оператора RENUMBER без параметров перенумеровывается вся программа, начиная с номера 10 с шагом 10. Этот оператор рекомендуется помещать первым в программе, которую нужно отлаживать.

Оператор RESTORE предназначен для организации чтения данных из оператора DATA, записанных с помощью оператора READ, в произвольной последовательности и нужное число раз. Например, в фрагменте программы

```
10 DATA 52, 54, 55, 58
20 RESTORE 3
30 READ X
40 PRINT /OC, X
```

оператором будет читаться третье значение (55), определенное оператором DATA, которое будет печататься на АЦПУ.

Оператор RETURN служит для обозначения конца подпрограммы и передачи управления оператору, следующему за оператором GOSUB или GOSUB' (вызов подпрограммы или помеченной подпрограммы). Посредством оператора RETURN CLEAR стирается адрес возврата из подпрограммы. Затем выполняется следующий за ним оператор.

Оператор RUN предназначен для запуска программы на выполнение. Оператором RUN ⟨ номер строки ⟩ запускается программа с указанной строки. Для повторного выполнения программы достаточно еще раз нажать клавишу RUN.

Оператор STOP служит для программированного останова во время выполнения программы. На операторе STOP выполнение программы приостанавливается и пользователь может произвести необходимые действия (подготовку и смену носителей, включение УВВ и т. д.). После нажатия клавиши CONTINUE программа продолжает выполняться, начиная с оператора, следующего за оператором STOP.

Оператор TRACE предназначен для пооператорной отладки программы. С его помощью пользователь может прослеживать ход выполнения программы. Режим трассировки реализуется от оператора TRACE до оператора TRACE OFF, при этом на системную консоль выводятся текст строки и результаты. Например, по фрагменту программы

```
10 TRACE
20 M = X + Y + Z : PRINT X, Y, Z, M
30 X = 5 : Y = 7 : Z = X ^ 2 + Y ^ 3
40 Y = 87 : PRINT / OC, Y
50 TRACE OFF
```

на АЦПУ выводится число 87, а на дисплей – 5, 7, 368 и 380.

Оператор ON ERROR дает пользователю возможность обрабатывать возникающие в программе ошибки. Он имеет следующую структуру:

```
ON ERROR [ A¤, B¤, GOTO ⟨H⟩ ]
```

Посредством оператора ON ERROR без данных отменяется программная обработка ошибок. При появлении ошибки в программе с помощью оператора ON ERROR код ошибки и номер строки, в которой появилась ошибка, помещаются соответ-

ственно в символьные переменные A¤ , B¤ , а управление передается оператору с номером H. Длина обеих символьных переменных по 4 байта и ее следует точно задавать при определении переменных (посредством оператора DIM). При наличии ошибок в первые 2 байта переменной A¤ заносится код, а в третий и четвертый байты – уточненное значение. Например, во фрагменте программы

```
120 DIM A¤ 4, B¤ 4, C¤ 22
.....
.....
150 ON ERROR A¤, B¤, GOTO 190
160 INPUT A, B
170 X = A/B
180 PRINT A, B, X
185 GOTO 160
190 C¤ = " ARITHMETICAL MISTAKE "
200 PRINT C¤, A¤, B¤, "REPEATE JOB"
210 GOTO 160
```

при возникновении ситуации (например, деления на нуль) в строке 170 в переменную A¤ заносится информация об ошибке (код 03), в переменную B¤ – значение 170 и управление передается строке 190, информирующей пользователя об ошибке и необходимости ее исправления.

Оператор GOTO ⟨H⟩ служит для передачи управления строке с номером H.

Например: 10 GOTO 140.

Оператор IF имеет следующую структуру:

IF I ⟨условие⟩ B THEN ⟨H⟩, т. е. если пара ⟨ I, B ⟩ удовлетворяет заданному условию, то управление получает строка с номером H, иначе выполняется строка, следующая за строкой с этим оператором. Оператор IF позволяет легко реализовать условные переходы в программе, но не может использоваться в строках НС. I и B могут быть как числовыми, так и символьными константами или переменными, но в одном операторе эти оба типа несовместимы. Например, во фрагменте программы

```
120 DIM A¤ 3
130 INPUT "INTRODUCE ANSWER (YES / NO)", A ¤
140 IF A¤ = "YES" THEN 170
150 IF A¤ = "NO" THEN 185
160 PRINT "MISTAKE IN ANSWER" : GOTO 130
170 PRINT A¤, "–YES"
180 GOTO 400
185 PRINT A¤, "–NO"
190 . . . . . .
```

в случае положительного ответа на дисплее печатается сам ответ, и управление передается строке 400, при наличии ошиб-

ки в ответе печатаются сообщения об ошибке и просьба снова повторить ответ, а в случае отрицательного ответа печатается ответ и работа продолжается со строки 190.

Оператор GOSUB ⟨H⟩ не используется в строках НС и не может быть последним в программе. Он предназначен для обращения к подпрограммам, идентифицируемым номерами Н их первых строк. Эти строки могут содержать любой оператор языка БЕЙСИК. Последним оператором подпрограммы должен быть оператор RETURN , с помощью которого осуществляется переход к основной программе – оператору, следующему за оператором GOSUB . Например, во фрагменте программы

```
100 A=5:B=6
110 GOSUB 130
115 GOTO 200
130 REM SUBROUTINE
140 C=A^2+B^2
150 PRINT/OC, C
160 RETURN
```

после выполнения подпрограммы, вычисляющей значение $C = A^2 + B^2$ с печатью полученного результата, управление передается строке 115.

Языком БЕЙСИК допускаются вложенные подпрограммы (когда в одной подпрограмме делается ссылка на другую и т. д.). Подпрограмма, организованная с помощью операторов GOSUB и RETURN , допускает обращение к ней по номеру любой своей строки. Оператор GOSUB не может быть последним в программе и служит для обращения к помеченной подпрограмме, определенной оператором DEFFN' [11]. Примеры на эти случаи приводятся ниже.

Оператор LET служит для присвоения значений цифровым или символьным переменным. Используя этот оператор, можно присвоить одно и то же значение сразу нескольким переменным. Например, LET A, B, C, T = 55; LET A ¤ , B ¤ = "SKB".

Оператор LET может использоваться и без своего имени: A ¤ , A1 ¤ = "МПСМ". В одном таком операторе нельзя совмещать цифровые и символьные переменные.

Оператор ON предназначен для организации перехода в программе и имеет следующую структуру:

ON ⟨AB⟩ GOTO ⟨H_1, H_2, ..., H_k⟩,

что по смыслу означает: значение p целой части арифметического выражения AB (p= [AB]) определяет номер H_p ($1 \leqslant p \leqslant k$) строки, которой будет передано управление; при $p > k$ или $p < 1$ выполняется строка программы, следующая за оператором

ON . Например, по фрагменту программы

```
10  INPUT A, B
20  ON A/B GOTO 50, 70, 90
30  PRINT "RESULTS" A, B
50  PRINT A ^ 2+B ^ 2, A, B
60  GOTO 10
70  PRINT A+B, A, B
80  GOTO 10
90  PRINT A*B, A, B
```

в зависимости от значения величины [A/B] будет осуществляться переход либо на строки с номерами 50, 70 и 90 (при $1 \leqslant [A/B] \leqslant 3$), либо на строку 30 (при $1 > [A/B] > 3$) при условии, что $B \neq 0$. По оператору ON можно также сразу выходить на подпрограмму. Например: 400 ON P GOSUB 5000, 6000, 7000.

Операторы FOR–TO и NEXT совместно позволяют организовать циклы в программе. Любой цикл состоит из трех частей:

1) заголовка (определяется оператором FOR – TO);

2) тела (содержит операторы БЕЙСИКа, допуская и другие циклы – вложенные);

3) конца (определяется оператором NEXT).

Оператор FOR–TO имеет следующую структуру:
FOR < A = P1 > TO < K > [STEP < M >], где первый параметр (A = P1) задает начальное значение P1 простой цифровой переменной A; второй (K) – определяет конечное значение переменной A; третий (M) – задает шаг изменения переменной A. При отсутствии записи STEP M шаг полагается равным 1. Отметим, что P1, K, M могут быть арифметическими выражениями. Конец цикла задается оператором NEXT A, где простая цифровая переменная A должна совпадать с переменной, заданной в операторе заголовка цикла FOR–TO.

Организация цикла имеет следующие особенности:

1) в цикл нельзя входить извне его;

2) допускается выход из цикла до его окончания.

Рассмотрим следующий простой пример оценки производительности ЦП:

```
1 INPUT M : INPUT ¤T1 : FOR A=1 TO M 1:M=10
2 A=A+ 1 : NEXT A : INPUT¤T2
                                  A=A+1:IF A<M THEN 2:
                                  PRINT "A="; A:STOP:
3 PRINT "A=";A, "T="; ROUND ((T1–T2)/2000,1):
                                  STOP:M= M*10:A= 0:GOTO2
4 GOTO 1
```

В зависимости от вводимого значения $M \in \{10^k / k = 2, 3, 4, \ldots\}$ получаем M-кратное выполнение оператора A = A + 1. Оценка времени выполнения такого цикла в определенной ме-

ре характеризует производительность процессора. Время подсчитывается автоматически таймером.

Второй цикл был организован проще: значение M каждый раз изменялось с пульта, подсчет времени хронометрировался секундомером. Для значений $M \in \{1, 2, 3, 4, 5\}$ была получена таблица временных затрат по обоим циклам (табл. 2).

Полученные результаты свидетельствуют о почти линейном возрастании времени вычислений и позволяют оценить производительность ЭВМ "Искра-226": не более 500–600 операторов БЕЙСИКа в секунду.

Оператор SELECT предназначен для выполнения следующих функций:

переопределения консолей;

задания адреса УВВ при выполнении операций ввода–вывода данных;

задания единицы тригонометрических функций;

задания продолжительности паузы при выводе информации на дисплей или АЦПУ.

Рассмотрим наиболее часто используемые функции, задаваемые оператором SELECT (остальные смотри в [10]):

SELECT C1	⟨ФАУ⟩	переопределяет адрес системного ввода;
SELECT CO	⟨ФАУ⟩	-"-
SELECT TAPE	⟨ФАУ⟩	переопределяет адрес системного УВВ на НМЛ;
SELECT DISK	⟨ФАУ⟩	-"- НМД;
SELECT LIST	⟨ФАУ⟩	определяет УВВ для вывода листинга программ;
SELECT PRINT	⟨ФАУ⟩	задает УВВ для вывода данных по операторам PRINT;
SELECT # ⟨Лн⟩	⟨[R/F]⟩	устанавливает соответствие между логическим номером (ЛН) и ФАУ.

Таблица 2. Временные затраты по обоим циклам, с

M	Цикл 1	Цикл 2
10	0.26	0.0
100	0.205	0.2
1000	2.015	2.0
10000	20.306	20.4
100000	197.351	197.0

Многочисленные примеры искользования этого важного оператора можно найти в приведенных ниже примерах программ. Оператор SELECT действует до выполнения одной из процедур:

выдачи другого оператора SELECT с другими данными;

использования оператора CLEAR без параметров (поскольку его желательно использовать перед каждой новой загрузкой программы, перед началом новой программы следует задавать, если нужно, новый оператор SELECT);

нажатия кнопки SR.

4.5. Операторы преобразования информации и функции символьных переменных

БЕЙСИК имеет ряд операторов, позволяющих изменять содержимое байтов данных. Эти операторы используются обычно при преобразовании данных и кодов, трансляции программ, приготовлении выходных данных в специальных форматах. Подробно они описаны в [10, 11, 18]. Здесь же, в табл. 3,

Таблица 3. Краткие сведения о символьных операторах и функциях

Оператор	Функция	Смысловое содержание
ADD		Сложение в бинарной форме логических аргументов
AND		Логическое умножение аргументов
BIN		Преобразование в бинарное число
BOOL		Выполнение любой из 16 возможных логических операций
CONVERT		Преобразование символьной переменной в цифровую и наоборот
INIT		Присвоение начальных значений символьным переменным и массивам
OR		Выполнение логической операции ИЛИ
PACK		Упаковка значений цифровых переменных и массивов в символьные по формату
UNPACK		Распаковка данных, упакованных по оператору PACK
ROTATE		Циклический сдвиг влево битов каждого байта символьной переменной
XOR		Выполнение логической исключающей операции ИЛИ
	LEN	Определение числа символов в переменной
	NUM	Определение количества символов в символьной переменной
	POS	Определение положения первого символа в заданной переменной
	VAL	Преобразование бинарного (шестнадцатиричного) значения первого символа переменной в десятичное число

приведены краткие сведения о символьных операторах и функциях, из которой следует, что БЕЙСИК обладает довольно широкими возможностями обработки символьной информации.

4.6. Операторы преобразования данных

Операторы языка БЕЙСИК позволяют преобразовывать данные, подготавливать их в форматах для непосредственной передачи, а также распаковывать полученные данные в форматы, используемые в ЭВМ "Искра-226". Как и в п.4.5, отразим эти возможности в табличной форме (табл. 4), отсылая за подробными сведениями к [15].

Таблица 4. Краткие сведения об операторах преобразования данных

Оператор	Смысловое содержание
¤ TRAN	Преобразование кодов.
¤ PACK	Последовательная упаковка значений переменных из списка аргументов в символьный аргумент, являющийся буфером для размещения данных
¤ UNPACK	Распаковка данных из символьного аргумента-буфера и последовательное присвоение распакованных значений переменным (массовым) из списка аргументов

На этом заканчиваем рассмотрение операторов обработки данных на языке БЕЙСИК и переходим к операторам ввода – вывода, которые позволяют работать с УВВ. Такие классы операторов, как матричные, сортировки (слияния), графики здесь не рассматриваются, а заинтересованного читателя отсылаем к [10, 12–14, 16].

4.7. Операторы ввода – вывода данных

Эти операторы предназначены для обмена информацией между программами пользователя и УВВ, а также другими устройствами (например, таймером). Для удобства рассмотрения разобьем их на три группы:

1) операторы для работы с НМЛ;
2) операторы для работы с НМД;
3) общие операторы ввода – вывода.

Операторы для работы с НМЛ и НМД во многом сходны, а так как практическая работа с НМЛ на выпускаемых до сих пор ЭВМ "Искра-226" исполнений 1–4 еще, как следует, не налажена, описание операторов для работы с НМЛ дадим в табличной форме. Подробные же сведения об этих операторах можно найти в [11].

Операторы для работы с НМЛ. В ЭВМ "Искра-226" используются НМЛ (аналогичные НМЛ, применяемым в СМ ЭВМ) с ФАУ 09 и встроенный кассетный НМЛ с ФАУ 08. Информация на НМЛ записывается файлами, состоящими из логических записей. Файл имеет структуру:

Заголовок файла	Программа и (или) данные	Конец файла

Заголовок файла, как правило, содержит имя файла и представляет собой физическую запись. Для программного файла она формируется автоматически, для файла данных – пользователем. Тело файла составляют программа и (или) данные. Конец файла представляет собой физическую запись, принцип формирования которой аналогичен заголовку файла. Программный файл состоит из одной логической записи, так как формируется одним оператором вывода.

Файловая организация на НМЛ позволяет пользователю организовать эффективную работу с данными и программами посредством операторов, указанных в табл. 5.

В качестве простого примера приведем фрагменты записи с защитой программы ПООСІ на НМЛ с ФАУ 09 и ее загрузку с этого же НМЛ:

```
10 REWIND /09
20 SAVE / 09, P "POOS1"
```

```
10 REWIND / 09
20 LOAD / 09, "POOS1"
```

Таблица 5. **Краткие сведения об операторах для работы с НМЛ**

Оператор	Смысловое содержание
BACK SPACE	Перемотка НМЛ назад (на К файлов логических или физических записей; К ≥ 1)
DATA LOAD	Чтение с НМЛ логических записей с присвоением их значений переменным и массивам
DATA LOAD BT	Чтение с НМЛ очередной физической записи с присвоением значения ее символьным переменной или массиву
DATA RESERVE	Обновление логических записей на НМЛ; можно менять все записи и имя файла
DATA SAVE	Занесение на НМЛ данных в виде логической записи либо в виде файла данных
DATA SAVE BT	Занесение на НМЛ данных без управляющей информации в виде физической записи
IF END THEN	Анализ на конец файла на НМЛ; по концу файла переход на заданную строку программы
LOAD	Загрузка программы или ее сегмента с НМЛ или из символьной обобщенной переменной (массива)
REWIND	Перемотка НМЛ и установка его в начало
SAVE	Запись на НМЛ программы или ее сегмента
SKIP	Перемотка НМЛ вперед

Общие операторы ввода – вывода. Операторы этой группы наиболее часто используются в программах пользователя и позволяют работать с клавиатурой, дисплеем, печатью, таймером и выполнять другие не менее важные функции по обработке данных. Число этих операторов невелико – восемь.

Оператор HEXPRINT служит для вывода на дисплей или АЦПУ символьной информации (переменных, массивов) в шестнадцатиричном коде. Например:

```
10 SELECT # 5 OC : DIM A¤ 14
15 A¤ = "SKB MPSM ESSR"
30 HEXPRINT A¤ : HEXPRINT #5, A¤
```

На АЦПУ и дисплей выводится информация:

```
534B42204D50534D204553535220
```

Оператор IMAGE обозначается символом (%) и предназначен для передачи оператору PRINTUSING описания формата в целях вывода информации на дисплей или АЦПУ. Оператор включает в себя текст для печати и форматы, по которым выводятся элементы, перечисленные в операторе PRINTUSING Он может содержать любые символы, кроме двоеточия, которые вставляются до и после описания формата. Описание формата должно иметь хотя бы один символ # или набор символов + –. //, ^ ; ¤ в определенной последовательности (ниже об этом будет сказано подробнее).
Например: 10% ГОД = #### МЕСЯЦ = ##ЧИСЛО = = ## .

Оператор (%) можно использовать и для ввода в программу текстов (комментариев) разного рода, которые впоследствии можно выводить оператором PRINTUSING . Например:

```
1% ВВОД ДАННЫХ ДЛЯ ВЕЛИЧИНЫ ⟨ X ⟩
2  PRINTUSING 1 : INPUT X : PRINT "X="; X
```

Оператор PRINTUSING служит для вывода цифровой и символьной информации на УВВ в требуемом формате, который задается символьной переменной, цепочкой символов в кавычках или оператором IMAGE (указанием номера строки с ним), причем оператор IMAGE может использоваться в нескольких операторах PRINTUSING .

Элементы PRINT в операторе PRINTUSING индуцируются согласно их очередности. Для распечатки числовых величин имеется три формата:

1) формат 1 для целых чисел, например (####);
2) формат 2 для действительных чисел, например (##.##);
3) формат 3 для чисел, записанных в экспоненциальной форме, например (#.# ^ ^ ^ ^).

При выводе цифровой информации каждое значение в операторе PRINTUSING корректируется согласно выбранному формату. Например, по фрагменту программы

```
10  A¤="    Размер    =#.###ВЕС=###"
 0  DIM A¤22:X=1.325:Y=524
30  PRINTUSING/OC,A¤,X,Y
```

на АЦПУ будет выведено:

РАЗМЕР = 1,325 ВЕС = 524,

а по фрагменту

```
10% ЧИСЛО=  ##  МЕСЯЦ=##  ГОД=  ####  СКБ
20  X=2:Y=1:M=1986
30  PRINTUSING/OC,10,X,Y,M
```

на АЦПУ появится строка: ЧИСЛО = 2 МЕСЯЦ = 1 ГОД = 1986 СКБ.

Подробные сведения о совместном использовании этих двух операторов можно найти в [10, 11].

Оператор KEYIN имеет структуру

KEYIN ⟨C11⟩,⟨H1⟩,⟨H2⟩

и предназначен для посимвольного получения информации с клавиатуры. Он функционирует следующим образом. Если клавиша нажата к моменту выполнения оператора, то ее шестнадцатиричный код помещается в первый байт символьной переменной и управление передается строке с номером H1. При нажатии одной из клавиш СФ ее код помещается в первый байт символьной переменной и управление получает строка с номером H2. Если к моменту выполнения оператора KEYIN клавиши нажаты не были, то он играет роль пустого оператора, т. е. выполняется следующий за ним оператор. Например, в процессе выполнения фрагмента программы

```
10  INPUT X¤
20  KEYIN X¤,30,50
30  PRINT "ЛАТЫНЬ",X¤
40  GOTO 10
50  PRINT "СФ",X¤
60  GOTO 10
```

на дисплее в зависимости от того, нажат ли латинский регистр или клавиши СФ, циклически будет появляться одно из сообщений: ЛАТЫНЬ ⟨клавиша⟩ или СФ ⟨клавиша⟩.

Оператор KEYIN позволяет организовать пользователю очень гибкие управление и обработку по инициативе самого пользователя, что особенно важно при выполнении длинных циклических конструкций.

Оператор LINPUT дает возможность вызвать значение заданной символьной переменной с последующим присвоением ей откорректированного значения. Длина вызываемого

значения ⩽ 240 байтов. Оператор позволяет вводить и корректировать символьные данные, включая пробелы, кавычки, запятые непосредственно в символьной переменной. Он имеет структуру:

LINPUT [⟨СК⟩,] ⟨СП⟩

При его выполнении значение символьной переменной СП заменяется на указанную символьную константу СК. Если первый параметр в символьной константе отсутствует, то на редактирование вызывается символьная переменная. Например, во фрагменте программы

```
10 DIM A¤ 10
20 A¤ ="PPPTOC-85"
30 LINPUT "RRTHS", A¤: PRINT A¤
```

содержимое A ¤ меняется на " RRTHS ". Если же во фрагменте строку 30 заменить на 30 LINPUT A ¤ , то на редактирование вызовется символьная переменная A ¤.

Оператор INPUT является одним из основных операторов ввода и позволяет вводить с клавиатуры цифровую и символьную информацию в процессе выполнения программы. По данному оператору можно также снимать показания таймера. При реализации оператора ЦП переходит в режим ожидания ввода информации с появлением знака вопроса (?) на дисплее. Данные следует вводить согласно списку переменных в операторе INPUT . Значение символьной переменной можно вводить в кавычках, апострофах или без них (если это допускает набор вводимых символов) [11].

Если введено меньше информации, чем указано в операторе, то снова появится знак вопроса и можно продолжать ввод. Если же нажать клавишу CR/LF , то управление получит следующий оператор. Например:

```
10  DIM A¤ 10, B¤ 10, C¤ 12, A (10)
20  INPUT "INTRODUCE INFORMATION", A¤
30  PRINT A¤: PRINT / OC, A¤
40  INPUT "INTRODUCE DATA", B¤, C¤: PRINT/ OC, B¤, C¤
50  INPUT "SYMBOLS AND NUMBERS", A¤, B¤, C¤, A(10)
60  PRINT / OC, A¤, B¤, C¤, A(10): END
```

В результате диалога с программой на дисплее и АЦПУ можно получить, например, следующие результаты:

```
14529
78956                183267
СКБ                  МПСМ              ЭССР        1986
```

С помощью оператора INPUT можно снимать показания таймера:

```
10  INPUT ¤T: PRINT T
```

Оператор LIST предназначен для вывода текста программы, находящегося в памяти ЭВМ, на дисплей или АЦПУ в порядке возрастания номеров строк.

Рассмотрим основные функции оператора, за деталями отсылая к [11]. Использование оператора LIST без параметров приводит к выводу на дисплей постранично (23 строки) всего текста программы. Но из-за большой скорости смены страниц такой режим особого смысла не имеет (хотя после ряда системных сбоев он применяется, локализуя программу). Поэтому следует применять оператор с параметром S, что приводит к выводу программы постранично, и для смены страниц следует поочередно нажимать клавишу CONTINUE. Можно, задавая начальный и конечный номера строк, выводить части программы. Примеры:

```
LIST : LIST S : LIST 10, 50 : LIST / OC : LIST / OC
```

Заметим, что защищенная программа посредством оператора LIST (ERR 44) не распечатывается и не совсем тривиально копируется на другой накопитель. Об этом подробнее говорится ниже.

Оператор PRINT служит для вывода цифровой и (или) символьной информации на дисплей либо АЦПУ. Рассмотрим его основные функции, за деталями отсылая к [11]. С помощью этого оператора выводятся символьные константы, переменные или массивы, а также цифровые константы, переменные или массивы. Оператор допускает совмещение обоих типов информации. Цифровые константы выводятся в естественной форме, если они находятся в пределах $10^{-1} - 10^{13}$, и в экспоненциальной – в противном случае. Функция TAB в операторе PRINT используется для вывода информации в определенное место строки (отсчет каждый раз идет с начала строки). Аргументом TAB может быть любое арифметическое выражение, значение которого должно лежать в пределах 0–225. Параметр AT в операторе предназначен для установки курсора на экране в заданное положение, им же можно корректировать информацию. Примеры:

```
PRINT/OC,A,B,C: RPINT "ТЕОРИЯ ОДНОРОДНЫХ СТРУКТУР"
PRINT AT(10, 20)        – установка курсора на строку 10, позицию 20
PRINT A ( ), C ¤ ( ) : PRINT AT (X, Y) : PRINT # 4, A * M+B, P
```

```
10  Y1= 80*RND(X) : Y2= 80*RND(X)
20  Z1= INT(Y1)-1 : Z2= INT(Y2)-1
30  PRINT AT(Z1, Z2) : PRINT/OC, AT(Z1, Z2)
40  GOTO 10: END
```

В приведенных примерах функция AT в операторе PRINT и функция псевдослучайных чисел RND позволяют случайным

образом организовать движение курсора по экрану и печатающей головки на АЦПУ. Функцию AT можно применить для организации программируемой печати на АЦПУ.

Операторы для работы с НМД. Выше рассматривались подготовка НМД к работе и режимы работы с ними: режим каталогов файлов и режим адресации секторов. Поскольку первый режим является основным для массового пользователя благодаря своей простоте и сервисному обслуживанию со стороны базового СПО, именно с операторами, используемыми в этом режиме, познакомимся подробнее. Что касается операторов, применяемых во втором режиме, их представим в табличной форме, отсылая заинтересованного читателя к [12]. Наряду с этим ниже приведен ряд примеров, иллюстрирующих применение дисковых операторов. Для краткости дальше будем использовать сокращение АВ – "арифметическое выражение".

Оператор SCRATCH DISK предназначен для резервирования секторов НМД под зону УК и его тело перед записью программ и файлов данных. Этот оператор должен выполняться до выполнения других операторов, используемых в режиме каталога файлов. Параметром LS=⟨AB⟩ оператора отводится под УК (начиная с нулевого) количество секторов НМД, равное целому значению АВ; по умолчанию принимается LS=24. Параметром END=⟨AB⟩ задается конец тела УК на НМД ([АВ] не превышает номера последнего сектора). Рекомендуется оставлять часть секторов на НМД следом за телом УК, чтобы иметь возможность размещать в них временные рабочие файлы. Примеры:

```
10 SCRATCH DISK R END=650     (отводится 24 сектора под
                               УК и 24–650 секторов под
                               его тело)
   SCRATCH DISK R      LS=2, END=990
```

Заметим, что среди сервисных средств ЭВМ "Искра-226" имеется ряд удобных программ ("Нива", A-SERVIS, PZP), позволяющих в диалоговом режиме не только подготавливать к работе любой магнитный накопитель (НМЛ, НМД, НГМД), но и существенно облегчать ведение файлового хозяйства.

Оператор MOVE END служит для изменения размера тела УК на НМД.

Структура его такова:

MOVE END {F/R} [⟨YBB⟩] = ⟨AB⟩,

где параметр ⟨AB⟩ определяет адрес нового конечного сектора тела УК. Информация при сжатии тела УК не разрушается. Примеры:

```
MOVE END R=600
MOVE END F#4, =1000-Y
```

Оператор DATA DC OPEN предназначен для резервирования секторов НМД под вновь создаваемый файл данных и занесения информации о нем в УК. Его можно применять для создания временных файлов за телом УК, используя параметр TEMP, или для повторного использования секторов файла, отмеченного как ликвидируемый. Каждый файл первоначально открывают, как правило, отдельным оператором

DATA SAVE DC OPEN

Параметр ¤ требует выполнения контрольного считывания после записи на НМД. Информация о временных файлах в УК не заносится.

Параметр ⟨номер файла⟩ указывает вновь создаваемый файл в ТУВВ (см. гл. 3). Параметр ⟨АВ⟩ служит для указания числа требуемых секторов под файл. Имя вновь создаваемого файла должно отличаться от имени файлов в УК, кроме случая, когда на место старого заносится новый файл. При создании временных файлов следует указывать число секторов большее на 1 для формирования операционной системой конца файла.

Пример:

250 DATA SAVE DC OPEN R(20) "ТАБЛИЦА" – на диске под файл "ТАБЛИЦА" отводится 20 секторов

300 DATA SAVE DC OPEN R #6 ("TAB") "ПООСІ" – на НМД с логическим номером 6 вместо файла "TAB" будет размещаться файл "ПООСІ"

340 DATA SAVE DC OPEN R TEMP, 800, 900 – на НМД с сектора 800 по сектор 900 будет записан временный рабочий файл

400 DATA SAVE DC OPEN R¤ (300) "ДАТА" – на 300 секторах НМД создается файл "ДАТА" и требуется контрольное чтение записи

Оператор DATA LOAD DC OPEN служит для открытия файлов (на запись или чтение), записанных ранее в каталог на НМД. При этом в ТУВВ заносятся номер файла и начальный адрес сектора. После открытия файла к нему можно обращаться посредством операторов режима каталога файлов, если номера файлов совпадают. По умолчанию данные о файле заносятся в нулевую строку ТУВВ. Параметр TEMP используется для открытия временного файла. Чтобы повторно открыть файл, всегда используется оператор DATA LOAD DC OPEN (даже при необходимости записи в него данных). Для обеспечения надежной работы рекомендуется при обращении к файлу каждый раз открывать его.

Примеры:

```
100 DATA LOAD DC OPEN R      "ДАТА"
120 DATA LOAD DC OPEN F #4, "РЕЗУЛЬТАТЫ"
130 DATA LOAD DC OPEN F TEMP 800, 850
```

Оператор DATA SAVE DC предназначен для записи данных из списка аргументов на НМД в виде одной логической записи. Он имеет структуру:

DATA SAVE DC[¤] ⟨номер файла⟩ { END / ⟨список аргументов⟩ }

Если номер файла отсутствует, то запись производится в файл с логическим номером 0, т. е. #0. Перед записью в файл его необходимо открыть. Параметр ¤ свидетельствует о необходимости контрольного считывания данных при их записи в файл. По параметру END в файл заносится запись окончания файла о количестве занятых секторов. После завершения каждой записи данных следует заносить и новую запись окончания файла. Если такая запись не будет сделана пользователем, то при новом обращении для записи в файл информация снова будет записываться с начала файла, т. е. на место старой информации, что иногда требуется. Примеры:

```
100 DATA LOAD DC OPEN R "ДАТА"
200 DIM A(3000)
300 DATA SAVE ¤DC A ( )
400 PRINT A ( ) : PRINT / OC, A ( ) : END
```

```
650 DATA SAVE DC ¤ # 3, A ( )
```

Оператор DSKIP служит для пропуска нужного числа логических записей (параметр ⟨AB⟩), если нет параметра S, или секторов – в противном случае. Если оператор содержит параметр END, то все логические записи пропускаются до записи конца файла. Оператор DSKIP организован так, что при его использовании выход за пределы файла невозможен. Структура этого оператора такова:

DSKIP [⟨номер файла⟩,] { END / ⟨AB⟩[S] }

Примеры:

```
100 DSKIP #6, END
120 DSKIP 42:DSKIP 47 S
```

Оператор DBACKSPACE предназначен для возврата назад на нужное число (заданное параметром ⟨AB⟩) логических записей, если нет параметра S, или секторов – в противном случае. Если оператор содержит параметр BEG, то происходит возврат на начало первой логической записи. При использовании этого оператора выход за пределы файла невозможен. Оператор DBACKSPACE имеет структуру:

DBACKSPACE [⟨номер файла⟩,] { ⟨AB⟩[S] / BEG }

Примеры:

```
100 DBACKSPACE #6, BEG
150 DBACKSPACE 100; 200 DBACKSPACE #1,35 S
```

Оператор DATA LOAD DS служит для чтения логических записей из каталога файлов с последующим присвоением считанных значений переменным (массивам) из списка аргументов. Перед чтением файл должен быть открыт посредством оператора DATA LOAD DC OPEN. Каждый оператор DATA LOAD DC обеспечивает считывание следующей за ним логической записи. Условие конца файла можно проверять с помощью оператора IF END THEN. Структура оператора DATA LOAD DC такова:

DATA LOAD DC [⟨Номер файла⟩ ,] ⟨список аргументов⟩

где параметр ⟨номер файла⟩ задает строку ТУВВ, содержащую информацию о файле. По умолчанию номер строки есть нуль. Действие оператора заканчивается после исчерпания списка аргументов. С помощью этого оператора можно читать только файлы данных, а не программ. Примеры:

```
DATA LOAD DC A0, S¤,G¤,T
DATA LOAD DC #3, A()
```

Оператор DATA SAVE DC CLOSE предназначен для закрытия всех открытых файлов, если задан параметр ALL, или файла, номер которого указан в операторе (по умолчанию закрывается файл с нулевым номером). Данный оператор рекомендуется использовать для закрытия файлов после завершения работы с ними, так как после его выполнения обеспечивается сохранность данных.

Примеры:

```
DATA SAVE DC CLOSE          – закрытие нулевого файла
DATA SAVE DC CLOSE ALL      – закрытие всех файлов
DATA SAVE DC CLOSE #6
```

Оператор LIMITS служит для получения пользователем информации и местоположении файла и его размерах. Подробные сведения об этом операторе имеются в [12].

Оператор SAVE DC предназначен для записи программ на НМД. С помощью этого оператора в УК заносятся имя файла, его тип (P/D), адреса начального и конечного секторов НМД. Оператор имеет структуру:

SAVE DC⟨тип диска⟩[¤] [УВВ,] [P/T/G] [AB имя 1] ⟨имя 2⟩ [⟨номер строки⟩] [⟨номер строки⟩],

где ⟨имя 1⟩ – имя старого (удаляемого) файла и ⟨имя 2⟩ – имя записываемой программы. Параметр ¤ задает режим контрольного чтения после записи программы на НМД. Параметр ⟨AB⟩ указывает на необходимость резервирования некоторого количества секторов сверх тех, которые занимает программа. Эти секторы можно использовать для последующего расширения программы.

По параметру ⟨имя 1⟩ новая программа "имя 2" записывается на место старой программы "имя 1"; [до этого файл "имя 1"

(программа или данные) должен быть помечен оператором SCRATCH или другим способом как удаляемый]. Параметр ⟨ имя 1 ⟩ и ⟨ имя 2 ⟩ могут совпадать. Обычно это имеет место в режиме отладки и корректировки программ. Если оба эти параметра отсутствуют, то новая программа (программа из ОП) записывается в свободной части тела каталога.

Параметры P, T, G задают признак защиты программы при записи на НМД (параметры P и G указывают на защиту от распечатки и повторной записи), параметры ⟨ номер строки ⟩ – начальную и конечную строки записываемой программы.

Примеры:

```
SAVE DC F T "POOS"
SAVE DC R¤T ("POOS") "POOS1"
SAVE DC R # 3, G ("PROG1") " POOS "
```

Оператор LOAD DC служит для загрузки программы с данным именем с НМД в ОП. Структура его такова: LOAD DC ⟨тип диска⟩ [⟨УВВ⟩ ,] ⟨имя⟩ [НС1] [, ⟨ НС2 ⟩] где параметры ⟨ тип диска ⟩ , ⟨ УВВ ⟩ задают адрес НМД с программой, ⟨ имя ⟩ – имя программы ⟨НС1⟩ и ⟨ НС2 ⟩ – номера строк загружаемого сегмента программы. Данный оператор позволяет пользователю удобно организовать работу с сегментированными программами. Подробные сведения об этом операторе можно найти в [12].

Примеры:

```
LOAD DC R "POOS"
LOAD DC F #3, "POOS1" 100, 420
```

Оператор VERIFY предназначен для проверки правильности записи информации на НМД. Он имеет очень простую структуру:

VERIFY ⟨тип диска⟩ [⟨УВВ⟩ ,] [(⟨ АВ1 ⟩ , ⟨ АВ2 ⟩)],

где параметры ⟨ АВ1 ⟩ , ⟨ АВ2 ⟩ задают начальный и конечный адреса секторов, проверяемой области НМД. При отсутствии этих параметров проверяется весь НМД (секторы 0–1000). В случае обнаружения ошибки выдается следующее сообщение:

ERROR IN SECTOR XXXX (XXXX – номер сектора).

Примеры:

```
VERIFY F; VERIFY R #5 (10, 114)
VERIFY R (0,1)
```

Оператор MOVE служит для копирования каталога с диска на диск с удалением всех ликвидируемых файлов из тела каталога и их имен из УК, оставшиеся файлы уплотняются в теле каталога и информация об этом соответственно корректируется в УК. Временные рабочие файлы не копируются. Оператор имеет структуру:

MOVE [⟨УВВ⟩,] { FR/RF }

(FR – при копии F→R, RF – при копии R→F), причем для

копирования необходимо выполнить следующие операции:

1) разметить НМД для записи посредством оператора SCRATCH DISK;

2) ФАУ диска, с которого делается копия, ввести с помощью оператора SELECT DISK;

3) в операторе MOVE ⟨УВВ⟩ задать диск, на который будет вестись запись.

Если параметр ⟨УВВ⟩ отсутствует, то его надо ввести, используя оператор SELECT DISK. Примеры:

```
10 MOVE RF: MOVE#3, RF
```

Оператор COPY предназначен для физического копирования информации с диска на диск. Структура его такова:

COPY [⟨УВВ⟩,][RF/FR][(⟨AB1⟩,⟨AB2⟩)],

причем параметр ⟨УВВ⟩ определяется так же, как и в операторе MOVE, а параметры ⟨AB1⟩ и ⟨AB2⟩ задают адреса секторов начала и конца копируемой области диска. В отличие от оператора MOVE оператор COPY при копировании каталога ликвидируемые файлы не удаляет.

Примеры:

```
10 COPY RF (0, 1000) : COPY RF
```

Оператор LIST DC имеет структуру:

LIST DC ⟨тип диска⟩ [⟨УВВ⟩,] [⟨символьная переменная или константа⟩]

и служит для распечатки содержимого УК на дисплее (или АЦПУ, заданном с помощью оператора SELECT LIST OC). Параметрами ⟨тип диска⟩ и ⟨УВВ⟩ выбирается тот диск, содержимое УК которого выводится. Пример фрагмента программы с использованием данного оператора:

```
REMOVABLE CATALOG
INDEX  SECTORS = 00003
END CAT. AREA  = 01000
CURRENT END    = 00323
NAME      TYPE     START      END      USED
CORRSECT    P       00003     00014    00006
PRINTSEC    P       00015     00022    00004
DATA        D       00023     00322    00001
```

На листинге идентифицируется следующая информация: размер (в секторах) УК, конец тела каталога, конец занятых секторов в теле каталога; имена всех файлов; типы файлов (D-данные, P – программа, SP или SD – удаляемые файлы); начало, конец файла и реально занимаемое им место в каталоге (в секторах). Параметр ⟨символьная переменная или константа⟩ позволяет просматривать каталог выборочно; при отсутствии этого параметра просматривается весь каталог. Например, по оператору

```
LIST  DC  FA¤="POOS"
```

распечатывается УК только для файлов, первые четыре буквы

имени которого есть POOS; по оператору LIST DC R – весь каталог консольного диска R.

Оператор SCRATCH имеет структуру:

SCRATCH ⟨тип диска⟩[⟨УВВ⟩,] ⟨список имен⟩

и предназначен для присвоения каталогизированным файлам с именами из ⟨списка имен⟩ статуса "ликвидируемые", т. е. они помечаются символом S. К таким файлам не имеют доступа операторы режима каталога. Подробнее статус файлов и его изменение будут рассматриваться ниже. Примеры:

SCRATCH R"ПООС", "ДАТА", "СКБ"

О том, как использовать временные рабочие файлы, уже говорилось выше. Отметим еще раз, что на них воздействуют операторами режима каталога файлов. Но так как им нельзя присваивать имена, то вместо имени файла применяется специальный параметр TEMP, который задается в операторе DATA SAVE DC OPEN при первом открытии временного файла [12].

Рассмотрим теперь режим адресации секторов НМД. В этом режиме используется восемь операторов. Пользователь не обеспечивается автоматическим ведением файловой службы, но может по своему усмотрению сам реализовать более эффективный (или специальный) для его целей режим работы с НМД. Для этого он должен обладать необходимыми квалификацией и навыками. Подробно операторы режима адресации секторов описаны в [12].

Смысловое содержание их раскрыто в табл. 6.

Примеры использования дисковых операторов в режиме адресации секторов отражают программы, помещенные ниже.

Таблица 6. Операторы режима адресации секторов НМД

Оператор	Смысловое содержание
DATA SAVE DA	Запись данных из списка аргументов на НМД, начиная с заданного сектора
DATA LOAD DA	Чтение логических записей с НМД, начиная с заданного сектора, с присвоением их значений переменным (массивам) из списка аргументов
DATA SAVE BA	Посекторная запись символьной информации объемом не более 256 байт на НМД
DATA LOAD BA	Считывание информации из одного сектора НМД в символьные переменную или массив
SAVE DA	Запись программы в заданное место НМД
LOAD DA	Загрузка с НМД в ОП программы или ее сегмента
COPY	Аналогичен такому же оператору режима каталога файлов
VERIFY	То же

4.8. Языки программирования БЕЙСИК 01 И БЕЙСИК 02

Как уже упоминалось выше, в настоящее время массовый пользователь работает с двумя операционными средами – языками БЕЙСИК 01 И БЕЙСИК 02, которые несовместимы между собой. Поэтому необходимо знать отличия их один от другого, тем более что с использованием этих операционных сред разработаны и продолжают разрабатываться прикладные программные средства. При этом все программные средства, разработанные на языке БЕЙСИК 01, надо перетранслировать на язык БЕЙСИК 02 и в ряде случаев соответственно изменить исходный текст программ.

Искушенный читатель, хорошо знакомый с языком БЕЙСИК 01, работая с программными материалами, замечает основные отличия обоих языков. Тем не менее, ниже приведена сводка основных отличий сравниваемых языков программирования, ориентированная на массового читателя. В ряде мест сводки читатель может заметить ее отличие от материалов технической документации на ЭВМ "Искра-226". Это связано с отсутствием полной адекватности (или ее неполнотой) технической документации и практической реализацией СПО ЭВМ "Искра-226".

Первую группу отличий языка БЕЙСИК 02 от языка БЕЙСИК 01 составляют изменения реализации операторов:

1. Оператор PRINT позволяет выводить на печать в естественной форме значения чисел X в пределах

$$1E-12 \leqslant |x| < 9.999999999999E+12.$$

2. В операторах CLEAR и LOAD исключена индикация ошибок при задании несуществующих программных строк, что дает возможность более гибко использовать эти операторы.

3. По оператору LOAD можно загружать программу из символьного аргумента с использованием переменной, не объявленной оператором COM.

4. Изменены коды HEX, формируемые по оператору KEYIN.

5. Изменена реализация оператора LINPUT, что позволяет более эффективно использовать его при корректировке текста программ.

6. В операторах ¤ TRAN, ¤ PACK, ¤ UNPACK, MATSEARCH, MATCOPY исключен параметр ⟨поле массива⟩.

7. В операторе SELECT DISK исключен параметр ⟨длина строки⟩ и указываемый адрес УВВ заносится в нулевую строку таблицы устройств системы (ТУС). По оператору вида SELECT # K указываемый адрес УВВ помещается в K-ю строку ТУС (K = 0 – 7); при этом предыдущая информация, содержащаяся в данной строке, стирается.

8. При указании в операторах ввода – вывода логического номера (# К) выбор ФАУ производится из К-й строки ТУС. При отсутствии указания УВВ в дисковых операторах ФАУ выбирается из нулевой строки ТУС.

9. Игнорируется использование оператора ON ERROR в строке НС.

10. В операторе LIST' исключен параметр ⟨номер строки⟩.

11. В дисковых операторах режима прямой адресации секторов может отсутствовать переменная, содержащая очередной адрес сектора.

Вторую группу отличий языка БЕЙСИК 02 от языка БЕЙСИК 01 составляют дополнительные операторы и расширение функций ряда имеющихся операторов:

1. Введен оператор REPLACE, обеспечивающий контекстные изменения символьных переменных.

2. Введен оператор ASMB, обеспечивающий работу с программами на языке Ассемблер-226.

3. Операторы DIM и COM допускают использование массивов объемом 64000 байтов.

4. Оператор COPY дополнен параметром, задающим адрес начального сектора НМД, на который будет производиться запись информации.

5. Операторы LOAD, LOAD DC и LOAD DA дополнены параметром, позволяющим использовать номер строки, с которой будет выполняться программа после ее загрузки в память, что существенно упрощает организацию многомодульных программ динамической структуры управления.

6. Оператор MATSEARCH дополнен параметром, позволяющим использовать операции отождествления и диапазона.

7. Оператор ROTATE дает возможность дополнительно осуществлять циклический сдвиг вправо битов каждого байта символьной переменной.

8. Функция VAL позволяет преобразовывать 2 байта символьного аргумента в числовое значение.

9. Оператор BIN дает возможность преобразовывать целые части АВ в бинарное число, помещаемое в первые 2 байта символьной переменной.

10. Оператор RETURN CLEAR дополнен параметром ALL, по которому сбрасывается весь магазин возвратов из циклов и подпрограмм.

11. Оператор LIMITS в форме с именем файла позволяет использовать ФАУ и четвертую числовую переменную, в которую заносится состояние файла.

12. Оператор STOP дает возможность индикации номера строки, где он используется.

13. В операторе IF – THEN могут быть логические выражения, содержащие логические AND, OR, XOR.

14. Оператором RESTORE можно ссылаться на номер строки, содержащей оператор DATA.

15. Оператор LIST позволяет выводить программные строки, содержащие заданный контекст и ТУС.

16. Оператор MOVE допускает указание второго адреса УВВ и возможность перезаписи файлов с диска на диск (выборочно).

17. Оператор HEXPRINT допускает использование символьной константы.

18. Функции NUM, LEN, POS и оператор INIT допускают использование параметров ⟨символьные массив и константы⟩.

19. Операторы ADD, AND, BOOL, OR, XOR допускают использование в первом параметре символьного массива, во втором – символьных массива и константы.

20. Оператор DATA допускает использование числового и символьного элементов.

21. Оператор PRINTUSING допускает использование символьного массива.

22. Операторы DATA LOAD DC OPEN, DATA SAVE DC OPEN допускают использование параметра ⟨ФАУ⟩; при этом заданный адрес, УВВ заносится в нулевую строку ТУС.

23. Оператор ON ERROR допускает использование параметров перехода GOSUB и THEN, что существенно расширяет возможности программной обработки особых и аварийных ситуаций во время выполнения программы.

24. В режиме редактирования (клавиша EDIT) добавлена возможность объединения нескольких строк в одну программную строку.

25. Посредством операторов UNPACK и ¤ UNPACK можно осуществлять проверку распаковываемых тетрад на наличие в них цифровых знаков.

26. Операторы LINPUT и INPUT при нажатой клавише HALT/STEP перестают функционировать до останова программы.

27. В режиме трассировки (оператор TRACE) имеется возможность вывода служебных символов в виде точек и вывода сообщений при выполнении оператора конца цикла NEXT.

28. Изменен синтаксис входного языка БЕЙСИК.

29. Изменены индикация готовности и аварийных ситуаций в операционной среде БЕЙСИКа.

Из вышеперечисленного следует, что БЕЙСИК 02 является существенным расширением языка БЕЙСИК 01. При этом, если для перевода программ, составленных для среды языка БЕЙСИК 01, в среду языка БЕЙСИК 02 практически (кроме перетрансляции) никаких переделок не требуется, то в случае обрат-

ного перевода (если такая необходимость возникает) может потребоваться, порой, значительная переделка программ с учетом различий обеих языковых сред.

Следовательно, практическая совместимость ПО в среде БЕЙСИКа имеется только сверху вниз: от языка БЕЙСИК 01 к языку БЕЙСИК 02.

Ниже рассматривается простой подход к расширению выразительных средств языка БЕЙСИК в среде самого этого языка.

4.9. Расширение возможностей выразительных средств языка БЕЙСИК

Выше рассмотрены средства программирования в операционной среде БЕЙСИКа и особенности их использования применительно к ЭВМ "Искра-226".

Приведенный материал убеждает нас в том, что средства языка БЕЙСИК с той или иной степенью выразительности и эффективности позволяют описать любой алгоритм обработки информации или вычислений, особенно в режимах диалога и отладки программ. Однако найдется не один пользователь, который при работе с ЭВМ "Искра-226" хотел бы располагать еще более выразительными средствами. При этом под выразительностью того или иного программного средства будем понимать его способность обеспечивать пользователя набором стандартных процедур или функций для описания часто используемых операторов обработки информации или вычислений. Например, матричные операторы языка БЕЙСИК дают возможность весьма эффективно и в компактной форме описывать операции линейной алгебры над матричными объектами.

Следовательно, чем выше уровень выразительности языка БЕЙСИК, тем меньше пользователю надо заниматься программированием решаемых им задач. Например, вычисление элементарных функций практически не требует дополнительно программирования.

С другой стороны, понятие выразительности программного средства в определенной мере относительно. Действительно, язык БЕЙСИК, располагая довольно выразительными средствами работы с матричными объектами, имеет слабые выразительные средства для работы на битовом уровне. Поэтому о выразительности программного средства целесообразно говорить относительно того или иного класса задач. Выразительность не связана тесно и с понятием возможностей программного средства: все универсальные языки программирования, обладая одинаковыми вычислительными возможностями, имеют при этом разный уровень выразительности относительно разных классов задач. Именно о расширении воз-

можностей выразительных средств языка БЕЙСИК будет идти речь ниже.

С этой целью предлагается один простой подход с использованием исключительно возможностей самого языка БЕЙСИК. Расширение его возможностей заключается в предоставлении пользователю новых, дополнительных операторов языка БЕЙСИК (табл. 7), которые описываются ниже.

Для обеспечения возможности работы с этими дополнительными операторами в среде БЕЙСИКа разработана специальная программа "Интерпретатор", которая позволяет любую исходную программу, составленную на языке БЕЙСИК и содержащую дополнительные операторы, переводить в эквивалентную ей программу на входном языке БЕЙСИК ЭВМ "Искра-226". Ниже даются краткая характеристика программы "Интерпретатор", ее листинг и рекомендации по использованию. Здесь же кратко описывается подход, положенный в основу реализации данной программы.

Этот подход применим к простым интерпретирующим программам, основными функциями которых являются:

перевод программ из одного диалекта языка БЕЙСИК на другой;

расширение возможностей выразительных средств языка БЕЙСИК;

Таблица 7. Дополнительные операторы языка БЕЙСИК

Оператор	Краткая характеристика
INTEGRAL	Вычисление определенного интеграла одной переменной
ROOT	-"- действительного корня уравнения
MODULL	-"- модуля числа по основанию P
MINMAX	-"- минимакса действительного числового массива заданной размерности
FUNCTION	-"- функции K переменных в данной точке
HEXTOTEN	Перевод шестнадцатиричного числа в десятичное
BINCOEFF	Вычисление биномиальных коэффициентов
INTERVAL	-"- местоположения двух заданных символов в указанной символьной переменной
CAPITALS	Работа с прописными и строчными буквами текста
MODIFY	Динамическая модификация текста программы в ОП
DISKNAME	Идентификация имени дискового тома
¤ DISKADR	Определение заданного НМД резидентным (консольным)
¤ BAEXCHD	Процедура обмена информацией в режиме прямой адресации типа ВА
¤ DAEXCHD	То же, типа ДА
¤ BITWORK	Реализация побитной работы с символьной информацией
¤ IF	Обобщенный оператор условного перехода
¤ ON	То же, вычисляемого перехода

создание в операционной среде БЕЙСИКа специализированных языковых процессов;

создание различных диалектов языка БЕЙСИК.

Пользователь, создав программу на языке БЕЙСИК, с использованием названных дополнительных операторов, вводит ее в память ЭВМ, игнорируя ошибки, указывающие на применение недопустимых операторов (подробные требования к программе будут раскрыты ниже). Затем с заданного НМД в ОП загружается программа "Интерпретатор" с именем "А – BASIC" и запускается на выполнение директивой RUN 9764.

В начале своего функционирования программа "Интерпретатор" выгружает программу пользователя в символьный массив 00 ¤ ( ) и очищает от нее память ЭВМ. Алгоритм интерпретации достаточно прост и состоит в следующем (рис. 4).

Программа "Интерпретатор", обрабатывая исходную программу пользователя, находящуюся в массиве 00 ¤ ( ), определяет ее длину и номер последующей текущей программной строки (операция 1). Затем осуществляется последовательный анализ программы пользователя с целью определения в ней дополнительных операторов, указанных в табл. 7. Если они в исходной программе отсутствуют, то интерпретация заканчивается, исходная программа остается без изменения в массиве 00 ¤ ( ) и программа "Интерпретатор" выходит на свое завершение с выдачей следующей основной информации:

время интерпретации в минутах;

значения длин исходной и выходной программ и декремента в процентах (относительное удлинение исходной программы);

запрос дальнейших действий пользователя (листинг выходной программы, запуск ее на выполнение и запись на НМД).

Встретив дополнительный оператор (операция 2), программа "Интерпретатор" на его месте в исходной программе формирует обращение к подпрограмме или помеченной подпрограмме GOSUB ⟨НС⟩ / GOSUB' ⟨НП⟩ [⟨параметры⟩] (операция 3). Если обращение было сформировано по оператору GOSUB, то согласно программе "Интерпретатор" на протяжении всего текста исходной программы аналогичные операторы заменяются данным обращением и только после этого может генерироваться само тело подпрограммы посредством оператора GOSUB', причем обращение формируется по такому принципу: если алгоритм, определяемый дополнительным оператором, от его параметров не зависит, то обращение будет по оператору GOSUB; в противном случае – по оператору GOSUB. Из 17 приведенных в табл. 7 дополнительных операторов только для шести формируется обращение по оператору GOSUB.

Генерация тела подпрограммы любого типа (за исключе-

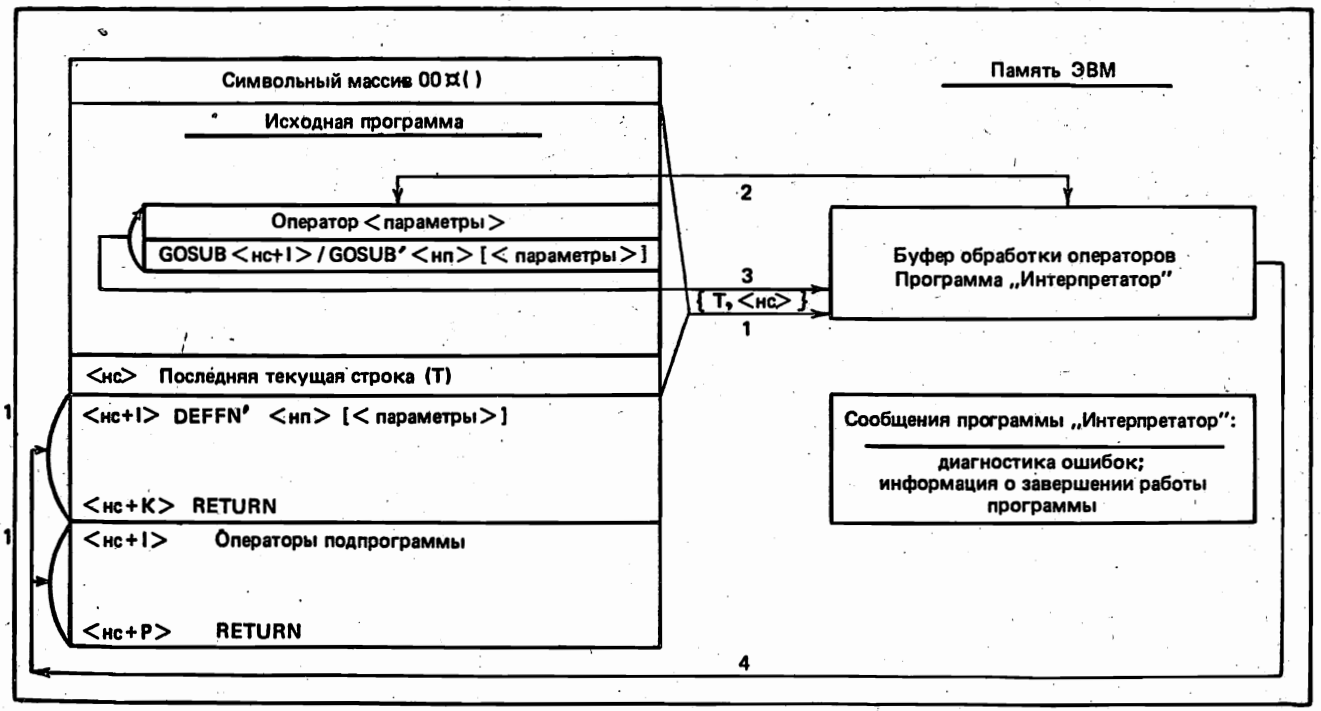

Рис. 4. Интерпретация исходной программы пользователя

нием очевидных различий) состоит в следующем. В соответствии с информацией о номере последней текущей программой строки (НС) и адресе (Т) ее конца в массиве 00 ¤ ( ) по программе "Интерпретатор" на основе параметров дополнительного оператора и соответствующего алгоритма генерируется тело подпрограммы, строки которого нумеруются с шагом 1, начиная с номера (НС + 1), и располагаются на свободных местах массива 00 ¤ ( ), начиная с (Т + 1)-й позиции (операция 4). После этого по программе "Интерпретатор" соответственным образом модифицируется информация о номере последней текущей строки и позиции ее конца в массиве 00 ¤ ( ) и осуществляется переход к интерпретации очередного дополнительного оператора.

После окончания интерпретации всей исходной программы программа "Интерпретатор" завершается так, как описано выше. В результате интерпретации формируется выходная программа на входном языке ЭВМ "Искра-226".

При генерации выходной программы в символьном массиве 00 ¤ ( ) выяснилось следующее интересное обстоятельство. В операционной среде БЕЙСИКа символьная информация, как известно, может вводиться без ограничителей, в кавычках, апострофах и с помощью HEX-функции. При модификации программ, выгруженных в символьные массивы по оператору SAVE, для последующего использования их после загрузки в память ЭВМ посредством оператора LOAD необходимо, чтобы корректировочная информация в программы из символьных массивов вводилась с помощью именно кавычек или HEX-функции, а не в апострофах, что уже обсуждалось выше.

Основные эксплуатационные характеристики программы "Интерпретатор" следующие:

размер объектного модуля – 11,5 Кбайта;

число операторов исходной программы – около 1000;

максимальное время интерпретации одного дополнительного оператора – 18с.

Размеры интерпретируемой программы (ИП) и символьного массива 00 ¤ ( ) программы "Интерпретатор" удовлетворяют следующему соотношению:

LEN (ИП) + LEN (00 ¤ ( )) ≤ 52000 байтов.

В самом начале функционирования программы "Интерпретатор" запрашивается объем памяти ЭВМ – LEN (00 ¤ ( )) для размещения в нем выходной программы. С учетом вышеизложенного этот объем нетрудно определить в каждом конкретном случае. Например: 5000, 10000, 40000 байтов.

Перейдем теперь к более подробному описанию дополнительных операторов языка БЕЙСИК.

Оператор INTEGRAL (A, B, E) S= f(X) служит для вычисления определенного интеграла от функции f (X) на заданном отрезке переменной (A, B) с точностью E. Результат его вычисления помещается в переменную S.

Оператор ROOT [f(x)] (A, B, E, S) предназначен для вычисления действительного корня функции f (x) на отрезке (A, B) с точностью E и начальным шагом S (при выполнении вычислений можно полагать S = E). Результат вычисления корня помещается в переменную X.

Оператор MINMAX Oo (R) = [N, M] служит для вычисления минимакса одномерного числового массива действительных чисел размерности R. Минимальное и максимальное значения массива присваиваются переменным N и M соответственно.

Оператор FUNCTION F (X_1, X_2, X_3, ... ,X_k) = f (X_1, X_2, X_3, ... , X_k) предназначен для вычисления значения функции k переменных в заданной точке. В качестве функций f (X_1, X_2, X_3, ... , X_k) могут быть только функции, определенные в операционной среде БЕЙСИКа, и конструкции из них.

Оператор HEXTOTEN (A¤)= N служит для перевода в десятичное число N шестнадцатиричного числа, заданного первыми двумя байтами символьной переменной A ¤.

Оператор BINCOEFF (K, N) = C предназначен для вычисления биномиальных коэффициентов бинома Ньютона $C = C_N^k$.

Оператор INTERVAL (A¤, B¤, I ¤) = [K1, K2] служит для определения местоположения [K1, K2] в символьной переменной I ¤ первого вхождения пары символов, определенных первыми байтами символьных переменных A ¤ и B ¤.

Оператор CAPITALS (A¤, P) [список номеров позиций] предназначен для определения букв переменной A ¤ в количестве P прописных букв согласно заданному списку номеров их позиций в символьной переменной A ¤.

Оператор MODIFY (N, A¤, B¤, M) служит для динамической модификации N-й строки программы заменой в ней контекста, содержащегося в переменной A ¤, на контекст из переменной B ¤ с последующей передачей управления программной строке с номером M.

Оператор DISKNAME (D ¤ , A ¤) = P предназначен для идентификации имени дискового тома, записанного в первых восьми байтах его последнего сектора. Параметр D ¤ определяет полный адрес НМД и параметр A ¤ – его имя. При совпадении имени заданного НМД, указанного в параметре A ¤ , с содержимым первых восьми байтов его последнего сектора переменная P получает значение 1, в противном случае – 0.

Оператор ¤ DISKADR (D ¤)〈T〉 служит для определения заданного полным адресом D¤ диска резидентным (консольным). После выполнения этого оператора обращаться к рези-

дентному диску можно только по параметру T (тип диска) в дисковых операторах. Оператор рекомендуется использовать перед первым выполнением дополнительного дискового оператора DISKNAME, ибо в противном случае осуществляется обращение к текущему резидентному диску. Данный оператор обязателен перед первым использованием дисковых операторов ¤ BAEXCHD и ¤ DAEXCHD.

Оператор ¤BAEXCHD(O,D¤,T)=A¤( ) предназначен для записи (0 = 1) в заданный параметром T сектор НМД, указанного параметром D ¤, символьной информации из массива A ¤( ). При O ≠ 1 по оператору осуществляется обратная операция – чтение символьной информации с НМД в массив A ¤( ).

Оператор ¤ DAEXCHD (O, D¤, T) ⟨ список переменных ⟩ ⟨ S ⟩ служит для записи (O = 1) в заданный параметром D¤ диск, начиная с сектора T, логической информации, определяемой параметром ⟨ список переменных ⟩ оператора. Переменная S получает значение номера, следующего за последним (использованным для записи) сектором диска. При 0 ≠ 1 по оператору производится обратная операция – чтение логической информации с указанного диска и присвоение ее значений переменным из списка оператора.

Оператор ¤ BITWORK (K, N, A ¤, 0) = B предназначен для обработки K-го бита N-го байта символьной переменной A ¤. При 0 = 1 в переменную B заносится значение искомого бита; при 0 ≠ 1 искомый бит в переменной A ¤ получает противоположное значение.

Оператор ¤ IF служит для организиции условного перехода в программе. Структура оператора такова:

¤ IF ⟨A⟩⟨ЛУ⟩ ⟨B⟩ { THEN ⟨ПС⟩ | PRINT [⟨список печати⟩] | GOSUB ⟨НС⟩ | GOSUB' ⟨НП⟩[⟨параметры⟩] | GOTO ⟨НС⟩ }

В случае выполнения логического условия (ЛУ) между арифметическими (символьными) выражениями A и B производится одно из следующих действий:

THEN ⟨ПС⟩ – выполняется указанная программная строка (ПС), отдельный оператор языка БЕЙСИК или группа операторов, разделенных символом & (символ THEN для обозначения этих операторов использовать запрещается);

PRINT – выводится на печать указанный список печати;

GOSUB – выполняется указанная подпрограмма;

GOSUB' – выполняется помеченная подпрограмма с заданным номером и передачей ей указанных параметров;

GOTO НС – передается управление ПС с НС.

После осуществления любого из этих действий (если в процессе их выполнения не было передач управления в другие места программы) выполняется следующий за ¤ IF оператор программы.

Оператор ¤ ON предназначен для организации обобщенного вычисляемого перехода в программе. Он имеет следующую структуру:

¤ ON(⟨ AB⟩) $[X_1]$ $[X_2]$ ... $[X_k]$,

где X_T – любые операторы языка БЕЙСИК или их объединения, разделенные символом &.

В зависимости от целой части T значения AB выполняются конструкция X_T $(T = \overline{1, k})$ и, если в ней нет передач управления в другие места программы, следующий за ¤ ON оператор программы. При $T \notin \{1, 2, ..., k\}$ действие оператора ¤ ON аналогично действию обычного оператора ON. Оператор ¤ON позволяет динамически модифицировать выполнение программы, передавая с возвратом управление ПС, отсутствующим в данный момент в программе.

Заметим, что все рассмотренные выше дополнительные операторы языка БЕЙСИК нельзя использовать в режиме непосредственного счета (НСЧ). Чтобы в программу, составленную на языке БЕЙСИК, можно было вводить дополнительные операторы, должны выполняться следующие основные требования:

1) дополнительные операторы должны кодироваться без ошибок согласно их синтаксису (включая и обозначения их переменных и параметров);

2) первое вхождение в программу операторов MODULL, MINMAX, HEXTOTEN, BINCOEFF, DISKNAME, ¤ BAEXCHD, ¤ BITWORK должно кодироваться полностью, во всех последующих вхождениях указывается только начальная часть их до знака равенства "=";

3) не рекомендуется использовать в программе следующие числовые и символьные переменные: Jo, Bo, Lo, Co, Po, Do, Eo, Qo, Ko, Xo; Ao¤, Bo¤, Co¤, No¤, Fo¤, Mo¤, Po¤, To¤, Ko¤, Zo¤, Go¤;

4) максимальный НС в программе не должен превышать 9763;

5) запрещается использовать для помеченных подпрограмм номера 0–2, 200–208;

6) запрещается использовать символьный массив Oo ¤ ( );

7) символьные переменные F ¤ 22, E ¤ 253, N ¤ 4, M ¤ можно применять только с указанными размерностями;

8) не допускается в общем случае рекурсивное использование дополнительных операторов.

Ниже приводится комплексный пример использования всех 17 дополнительных операторов и листинг выходной программы – результат обработки его программой "Интерпретатор". Этот листинг иллюстрирует вышеописанный принцип, лежащий в основе реализации программы "Интерпретатор", а также алгоритмы, описывающие функционирование дополнительных операторов. ПС 10–570 выходной программы соответствуют строкам 10–570 комплексного примера (исходной программы) с тем отличием, что в ней все дополнительные операторы заменены соответственно обращениями по GOSUB или GOSUB', а ПС 581–659 являются расширением исходной программы и сгенерированы они для обеспечения выполнения дополнительных операторов в среде БЕЙСИКа.

```
10 REM КОМПЛЕКСНЫЙ ПРИМЕР (ИНТЕРПРЕТАТОРА):SELECT PRINT0C(128)
20 DIM A¤(256)1,I¤25,O0(100):X=42:Y=47:Z=100:PRINT X,Y,Z
30 ¤IF X*Y>Z THEN Z=SQR(X+Y):PRINT Z:D¤='18F':¤DISKADR (D¤)(T)
40 ¤IF Z<10AND(X+Y)>Z PRINT (X+Y)/Z:STR(A¤(),1,6)='SKB087':O=1
50 T=1000:¤BAEXCHD (O,D¤,T)=A¤():INIT (HEX(20))A¤():O=2:T=1000
70 ¤BAEXCHD (O,D¤,T):¤IF STR(A¤(),1,6)='SKB087' GOSUB 300:A¤='SKB087'
80 DISKNAME (D¤,A¤)=P:T=997 :¤IF P=1 GOSUB ' 100(X,Y,Z,A¤):P=4
90 X=19773:MODULL (X,P)=Y:PRINT Y:X=424767:MODULL (X,P):PRINT Y
100 ¤IF Y^3<8 GOTO 110:PRINT "КОНЕЦ ПРОВЕРКИ ОПЕРАТОРА (IF)"
110 K=7:N=10:BINCOEFF (K,N)=C:PRINT C:K=5:BINCOEFF (K,N):PRINT C
120 FOR K=1TO100:O0(K)=K^2+15:NEXT K:R=50:MINMAX O0(R)=[N,M]
130 PRINT N,M:R=80:MINMAX O0(R):PRINT N,M:X=1:Y=5:Z=6:Q=4
140 FUNCTION F(X,Y,Z,Q)=X^2+Y^2+Z/(X+Y)+SQR(Q):PRINT "F=";F
150 FUNCTION F(X,Y,Z)=SQR(X^2+Y^2+Z^2)+X*Y*Z:PRINT "F=";F
160 A¤="[":B¤="]":I¤="11111[2222(333555]666)999":IF F>42THEN140
165 INTERVAL (A¤,B¤,I¤)=[K1,K2]:PRINT K1,K2,I¤:A¤="(":B¤=")"
170 INTERVAL (A¤,B¤,I¤):PRINT "K1=";K1;"K2=";K2:FOR K=1 TO 4
180 ¤ON(K)[PRINT K][U=1942&Z=56][GOSUB ' 120(K)][GOSUB 430]:NEXT K
185 FOR P=1TO2:¤ON(P)[LIST %][GOTO 190]:NEXT P
190 PRINT "K+1=";K+1;"U=";U:A¤="1947":HEXTOTEN (A¤)=N:PRINT A¤,N
200 A¤=HEX(1A0E):HEXTOTEN (A¤):PRINT A¤,N:A=1:B=2:E=0.01
210 INTEGRAL (A,B,E)S=X^2+5:PRINT "S=";S:A=2:B=3:E=0.1
215 INTEGRAL (A,B,E)S=EXP(X)+4:PRINT "S=";S:A=1:B=2:E,S=0.01
220 ROOT [X^4-3*X^3+3*X^2-2](A,B,E,S):PRINT "RUUT =";X:A=0:B=1
221 ROOT [X^3-1.26*X^2+0.5292*X-0.0741](A,B,E,S):PRINT X:GOTO 500
300 PRINT "ИМЯ ТОМА ";D¤;" ЕСТЬ ";STR(A¤(),1,6):RETURN
400 DEFFN ' 100(X,Y,Z,A¤):O=1:¤DAEXCHD (O,D¤,T)<X,Y,Z,A¤>(S)
410 PRINT S:O=2:¤DAEXCHD (O,D¤,T)<X,Y,Z,A¤>:PRINT X,Y,Z,A¤,S
420 RETURN :DEFFN ' 120(K):PRINT "K9=";K:PRINT "SKB MPSM":RETURN
430 PRINT "THEORY OF HOMOGENEOUS STRUCTURES":RETURN
500 A¤=HEX(0102030405060708090A15568932AAFF):HEXPRINT A¤:O=1:N=16
510 FOR K=1TO8:¤BITWORK (K,N,A¤,O)=B:PRINT "K=";K;"N=";N;"B=";B
520 ¤IF O=2 PRINT A¤:NEXT K:IF O=2THEN530:O=2:N=10:GOTO 510
530 N=540:A¤="1111111111111111":B¤="9999999999999999":M=550
540 PRINT "X=1111111111111111":LIST /0C,540:MODIFY (N,A¤,B¤,M)
550 LIST /0C,540:A¤="ABCDEFGNZRP[@*WV":HEXPRINT A¤
560 P=7:CAPITALS (A¤,P)[1,3,7,8,9,15,16]:HEXPRINT A¤
570 ¤IF P=7 THEN P=P^3&L=X^3&PRINT P,L:END :% КОНЕЦ ТЕСТИРОВАНИЯ
580 % ####################################################################
```

```
10 REM КОМПЛЕКСНЫЙ ПРИМЕР (ИНТЕРПРЕТАТОРА):SELECT PRINT0C(128)
20 DIM A¤(256)1,I¤25,O0(100):X=42:Y=47:Z=100:PRINT X,Y,Z
30 GOSUB 581:PRINT Z:D¤='18F':GOSUB ' 206(D¤)
40 GOSUB 589:STR(A¤(),1,6)='SKB087':O=1
50 T=1000:GOSUB ' 203(O,D¤,T):INIT (HEX(20))A¤():O=2:T=1000
70 GOSUB ' 203(O,D¤,T):GOSUB 595:A¤='SKB087'
80 GOSUB ' 207(D¤,A¤):T=997:GOSUB 601:P=4
90 X=19773:GOSUB ' 0(X,P):PRINT Y:X=424767:GOSUB ' 0(X,P):PRINT Y
100 IF Y^3<8THEN110:PRINT "КОНЕЦ ПРОВЕРКИ ОПЕРАТОРА (IF)"
110 K=7:N=10:GOSUB ' 204(K,N):PRINT C:K=5:GOSUB ' 204(K,N):PRINT C
120 FOR K=1TO100:O0(K)=K^2+15:NEXT K:R=50:GOSUB ' 200(R)
130 PRINT N,M:R=80:GOSUB ' 200(R):PRINT N,M:X=1:Y=5:Z=6:Q=4
```

```
140 GOSUB 609:PRINT "F=";F
150 GOSUB 610:PRINT "F=";F
160 A¤="[":B¤="]":I¤="11111[2222(333555]666>999":IF F>42THEN140
165 GOSUB ' 1(A¤,B¤,I¤):PRINT K1,K2,I¤:A¤="(":B¤=">"
170 GOSUB ' 1(A¤,B¤,I¤):PRINT "K1=";K1;"K2=";K2:FOR K=1TO4
180 ON (K)GOSUB615,616,617,618:NEXT K
185 FOR P=1TO2:ON (P)GOSUB619,620:NEXT P
190 PRINT "K+1=";K+1;"U=";U:A¤="1947":GOSUB ' 201(A¤):PRINT A¤,N
200 A¤=HEX(1A0E):GOSUB ' 201(A¤):PRINT A¤,N:A=1:B=2:E=0.01
210 GOSUB 623:PRINT "S=";S:A=2:B=3:E=0.1
215 GOSUB 626:PRINT "S=";S:A=1:B=2:E,S=0.01
220 GOSUB 629:PRINT "RUUT =";X:A=0:B=1
221 GOSUB 631:PRINT X:GOTO 500
300 PRINT "ИМЯ ТОМА ";D¤;" ЕСТЬ ";STR(A¤(),1,6):RETURN
400 DEFFN ' 100(X,Y,Z,A¤):O=1:GOSUB 633
410 PRINT S:O=2:GOSUB 636:PRINT X,Y,Z,A¤,S
420 RETURN :DEFFN ' 120(K):PRINT "K9=";K:PRINT "SKB MPSM":RETURN
430 PRINT "THEORY OF HOMOGENEOUS STRUCTURES":RETURN
500 A¤=HEX(0102030405060708090A15568932AAFF):HEXPRINT A¤:O=1:N=16
510 FOR K=1TO8:GOSUB ' 202(K,N,A¤,O):PRINT "K=";K;"N=";N;"B=";B
520 GOSUB 642:NEXT K:IF O=2THEN530:O=2:N=10:GOTO 510
530 N=540:A¤="1111111111111111":B¤="9999999999999999":M=550
540 PRINT "X=1111111111111111":LIST /0C,540:GOSUB ' 2(N,A¤,B¤,M)
550 LIST /0C,540:A¤="ABCDEFGNZRP[@*WV":HEXPRINT A¤
560 P=7:GOSUB ' 208(A¤,P):HEXPRINT A¤
570 GOSUB 657:END :% КОНЕЦ ТЕСТИРОВАНИЯ
580 % ###########################################################
581 IF X*Y>ZTHEN582:GOTO 583
582 Z=SQR(X+Y)
583 RETURN
584 DEFFN ' 206(D¤):IF D¤='1CR'THEN585:IF D¤='1CF'THEN586:IF D¤='18R'THEN587:SEL
ECT DISK18F:GOTO 588
585 SELECT DISK1CR:GOTO 588
586 SELECT DISK1CF:GOTO 588
587 SELECT DISK18R
588 RETURN
589 IF Z<10AND(X+Y)>ZTHEN590:GOTO 591
590 PRINT (X+Y)/Z
591 RETURN
592 DEFFN ' 203(O,D¤,T):GOSUB ' 206(D¤):IF O=1THEN593:DATA LOAD BA T(T)A¤():GOTO
 594
593 DATA SAVE BA T(T)A¤()
594 RETURN
595 IF STR(A¤(),1,6)='SKB087'THEN596:GOTO 597
596 GOSUB 300
597 RETURN
598 DEFFN ' 207(D¤,A¤):P=0:IF STR(D¤,1,2)<>'1C'THEN599:P=8791
599 DATA LOAD BA T(1000+P)B0¤:P=0:IF STR(B0¤,1,8)<>A¤THEN600:P=1
600 RETURN
601 IF P=1THEN602:GOTO 603
602 GOSUB ' 100(X,Y,Z,A¤)
603 RETURN
604 DEFFN ' 0(X,P):Y=X/P-INT(X/P):Y=ROUND(Y*P,0):RETURN
605 DEFFN ' 204(K,N):C=1:FOR J0=0TOK-1:C=C*(N-J0)/(J0+1):NEXT J0:C=ROUND(C,0):RE
TURN
606 DEFFN ' 200(R):M,N=O0(1):MAT REDIM O0(R):FOR J0=1TOR:IF O0(J0)>=NTHEN607:N=O
0(J0)
607 IF O0(J0)<=MTHEN608:M=O0(J0)
608 NEXT J0:RETURN
609 F=X^2+Y^2+Z/(X+Y)+SQR(Q):RETURN
610 F=SQR(X^2+Y^2+Z^2)+X*Y*Z:RETURN
611 DEFFN ' 1(A¤,B¤,I¤):FOR J0=1TOLEN(I¤):IF STR(I¤,J0,1)<>A¤THEN612:K1=J0
612 IF STR(I¤,J0,1)<>B¤THEN613:K2=J0:GOTO 614
613 NEXT J0
614 RETURN
615 PRINT K:RETURN
616 U=1942:Z=56:RETURN
617 GOSUB ' 120(K):RETURN
618 GOSUB 430:RETURN
619 LIST %:RETURN
620 GOTO 190:RETURN
621 DEFFN ' 201(A¤):N=1:GOSUB 622:B0=L0:C0=P0:N=2:GOSUB 622:D0=L0:E0=P0:N=B0*16^
3+C0*16^2+D0*16+E0:RETURN
622 N=VAL(STR(A¤,N,1)):L0=INT(N/16):P0=ROUND((N/16-L0)*16,0):RETURN
623 S,Q0=0:P0=0.1
624 FOR X=ATOB-P0STEPP0:S=S+(X^2+5)*P0:NEXT X:IF ABS(S-Q0)<=ETHEN625:Q0=S:P0=P0/
10:S=0:GOTO 624
625 RETURN
626 S,Q0=0:P0=0.1
```

```
627 FOR X=A%OB-P%STEPP%:S=S+(EXP(X)+4)*P%:NEXT X:IF ABS(S-Q%)<=ETHEN628:Q%=S:P%=
P%/10:S=%:GOTO 627
628 RETURN
629 FOR X=A%OBSTEPS:IF ABS(X^4-3*X^3+3*X^2-2)<=ETHEN630:NEXT X:S=S/10:GOTO 629
630 RETURN
631 FOR X=A%OBSTEPS:IF ABS(X^3-1.26*X^2+0.5292*X-0.0741)<=ETHEN632:NEXT X:S=S/10
:GOTO 631
632 RETURN
633 % %DAEXCH(O,D%,T):GOSUB ' 206(D%):IF O=1THEN634:DATA LOAD DA T(T,S)X,Y,Z,A%:
GOTO 635
634 DATA SAVE DA T(T,S)X,Y,Z,A%
635 RETURN
636 % %DAEXCH(O,D%,T):GOSUB ' 206(D%):IF O=1THEN637:DATA LOAD DA T(T,S)X,Y,Z:GOT
O 638
637 DATA SAVE DA T(T,S)X,Y,Z
638 RETURN
639 DEFFN ' 202(K,N,A%,O):B=0:BIN(N0%)=2^(8-K):H0%=N0%:AND(STR(N0%,1,1),STR(A%,N
,1)):IF STR(N0%,1,1)=HEX(00)THEN640:B=1
640 ON OGOTO641:BOOL 6(STR(A%,N,1),H0%)
641 RETURN
642 IF O=2THEN643:GOTO 644
643 PRINT A%
644 RETURN
645 DEFFN ' 2(N,A%,B%,M):DIM A0%253,F0%21,C0%26:F0%="HHHHSAVEA0%XXXX,GGGG"
646 C0%="NNN&LOADA0%TTTT,EEEE,KKKK":STR(F0%,21),STR(C0%,26)=HEX(85)
647 CONVERT NTOH0%,(####):CONVERT HTOH0%,(####):STR(F0%,1,4)='0649'
648 STR(F0%,12,4),STR(F0%,17,4)=N0%:STR(F0%,16,1)=',':LOAD F0%649,649,649
650 REPLACE L0,A0%,A%,B%:%ЗАМЕНА:STR(C0%,12,4),STR(C0%,17,4)=N0%:STR(C0%,22,4)=M
0%:STR(C0%,1,4)='0651':LOAD C0%651,651,651
652 RETURN
653 DEFFN ' 208(A%,P):DATA 1,3,7,8,9,15,16:FOR J0=1TOLEN(A%):FOR K0=1TOP:RESTORE
 K0:READ X0:IF J0()X0THEN655
654 P0%=HEX(40):T0%=HEX(50):K0%=HEX(E0):Z0%=HEX(F0):G0%=STR(A%,J0,1):BOOL 6(P0%,
G0%):BOOL 6(T0%,G0%):BOOL 6(K0%,G0%):BOOL 6(Z0%,G0%):IF VAL(P0%)>15ANDVAL(T0%)>1
5ANDVAL(K0%)>15ANDVAL(Z0%)>15THEN656:BOOL 6(STR(A%,J0,1),HEX(20)):GOTO 656
655 NEXT K0
656 NEXT J0:RETURN
657 IF P=7THEN658:GOTO 659
658 P=P^3:L=X^3:PRINT P,L
659 RETURN

                (ИНТЕРПРЕТАТОР) ПРЕОБРАЗОВАЛ ВАШУ
                ПРОГРАММУ ДЛЯ РАБОТЫ В (BASIC 02)
                ВРЕМЯ ИНТЕРПРЕТАЦИИ ПРОГРАММЫ = 6.1  МИН
ДЛИНА ИСХОДНОГО МОДУЛЯ = 2381 * ВЫХОДНОГО = 5644 * ДЕКРЕМЕНТ = 137  %
                ВОЗМОЖНЫЕ ТИПЫ РАБОТ С ПРОГРАММОЙ :
                ЛИСТИНГ(1),НА ДИСК(2),ЗАПУСК(3) ?
```

В заключение приведем служебную информацию, которую получает пользователь после завершения обработки исходной программы (выводятся на печать время интерпретации программы, длины исходного и выходного модулей в байтах, декремент выходной программы и дальнейшие виды работ с нею).

Заметим, что подавляющее большинство ограничений (в общем случае не столь существенных) вызвано тем, что при разработке программы "Интерпретатор" прежде всего ставилась задача продемонстрировать возможности операционной среды БЕЙСИКа по реализации подобного типа программных средств, а не создать развитое интерпретирующее программное средство универсального назначения. Такую задачу, интересную в практических целях, оставляем на разрешение читателям, советуя им внимательно и регулярно следить за разделом "Человек и компьютер" журнала "Наука и жизнь". Ведь даже рассмотренную здесь реализацию программы "Интерпретатор", на наш взгляд можно улучшить и сделать более насыщенной по описанным ее функциям.

На приведенном ниже листинге программы "Интерпретатор" можно еще раз проследить за использованием наиболее важных средств языка БЕЙСИК, позволяющих в самой среде

```
9764 REM (АЛАДЬЕВ#СКБ):PRINT HEX(03):DIM F%22,U%5,E%253,N%4,M%4:GOSUB ' 10:PRINT
9765 F%="9767 DIM O0%(HHHHH)1":STR(F%,22)=HEX(85):INPUT "ОПРЕДЕЛИТЬ РАЗМЕР ПАМЯТИ",N
9766 CONVERT NTOU%,(00000):STR(F%,14,5)=U%:LOAD F%9767,9767,9767:ON ERROR N%,M%GOTO9991
9768 INIT (' ')O0%():SAVE O0%(0,9763:CLEAR P0,9763:N,L=0:INPUT %A0
9769 K=LEN(O0%()):T0=K:T=K:GOTO 9770:DEFFN ' 0:U=VAL(W%,2):RETURN
9770 FOR K=T-1TO1STEP-1:IF O0%(K)=HEX(85)THEN9771:NEXT K:K=0
9771 L=K+1:FOR K=LTOL+3:IF O0%(K)=" "THEN9772:N=N+1:NEXT K
9772 CONVERT STR(O0%(),L,N)TON:PRINT HEX(0312):K9=1
9773 FOR K=K9TOT0-2:PRINT AT(8,1):PRINT ,"ИДЕТ ИНТЕРПРЕТАЦИЯ ВАШЕЙ ПРОГРАММЫ ДЛЯ"
9774 PRINT AT(12,1):PRINT ,"ИСПОЛЬЗОВАНИЯ ЕЕ В СРЕДЕ (BASIC 02) !"
9775 IF STR(O0%(),K,3)()"%IF"THEN9776:STR(O0%(),K,1)=" ":K=K+1:GOSUB ' 42(K)
9776 IF STR(O0%(),K,6)()"MODULL"THEN9777:GOSUB ' 43(K):GOSUB ' 87
9777 IF STR(O0%(),K,8)()"INTERVAL"THEN9778:GOSUB ' 44(K):GOSUB ' 87
9778 IF STR(O0%(),K,8)()"FUNCTION"THEN9779:GOSUB ' 45(K):GOSUB ' 87
9779 IF STR(O0%(),K,3)()"%ON"THEN9780:GOSUB ' 46(K):GOSUB ' 87
9780 IF STR(O0%(),K,8)()"%DAEXCHD"THEN9781:GOSUB ' 48(K):GOSUB ' 87
9781 IF STR(O0%(),K,8)()"%EXTOTEN"THEN9782:GOSUB ' 49(K):GOSUB ' 87
9782 IF STR(O0%(),K,8)()"INTEGRAL"THEN9783:GOSUB ' 50(K):GOSUB ' 87
9783 IF STR(O0%(),K,8)()"BINCOEFF"THEN9784:GOSUB ' 51(K):GOSUB ' 87
9784 IF STR(O0%(),K,8)()"%DAEXCHD"THEN9785:GOSUB ' 52(K):GOSUB ' 87
9785 IF STR(O0%(),K,8)()"%DISKADR"THEN9786:GOSUB ' 53(K):GOSUB ' 87
9786 IF STR(O0%(),K,6)()"MINMAX"THEN9787:GOSUB ' 47(K):GOSUB ' 87
9787 IF STR(O0%(),K,4)()"ROOT"THEN9788:GOSUB ' 54(K):GOSUB ' 87
9788 IF STR(O0%(),K,6)()"MODIFY"THEN9789:GOSUB ' 55(K):GOSUB ' 87
9789 IF STR(O0%(),K,8)()"DISKNAME"THEN9790:GOSUB ' 56(K):GOSUB ' 87
9790 IF STR(O0%(),K,8)()"%BITWORK"THEN9791:GOSUB ' 60(K):GOSUB ' 87
9791 IF STR(O0%(),K,8)()"CAPITALS"THEN9792:GOSUB ' 61(K):GOSUB ' 87
9792 NEXT K:PRINT HEX(030712):PRINT ,"(ИНТЕРПРЕТАТОР) ПРЕОБРАЗОВАЛ ВАШУ"
9793 PRINT AT(4,1):PRINT ,"ПРОГРАММУ ДЛЯ РАБОТЫ В (BASIC 02)"
9794 INPUT %A1:PRINT AT(8,1):PRINT ,"ВРЕМЯ ИНТЕРПРЕТАЦИИ ПРОГРАММЫ =";
9795 PRINT ROUND((A0-A1)/120000,1);" МИН":T1=LEN(O0%())
9796 PRINT "ДЛИНА ИСХОДНОГО МОДУЛЯ =";T0;"# ВЫХОДНОГО =";T1;
9797 PRINT "# ДЕКРЕМЕНТ =";ROUND(T1#100/T0-100,0);" %"
9798 IF T1-T0)=0THEN9799:PRINT ,"ОШИБКА В ИНТЕРПРЕТАЦИИ ПРОГРАММЫ !":STOP
9799 PRINT AT(14,1):PRINT ,"ВОЗМОЖНЫЕ ТИПЫ РАБОТ С ПРОГРАММОЙ :"
9800 PRINT AT(18,1):PRINT ,"ЖИСТИК"(1),НА ДИСК(2),ЗАПУСК(3) ?"
9801 INPUT "ОТВЕТИТЬ СООТВЕТСТВУЮЩИМ НОМЕРОМ",P0:LOAD O0%()0,0,9993
9802 DEFFN ' 42(K):INIT (HEX(20))E%:M=0:FOR P=KTOK+253
9803 IF STR(O0%(),P,1)="%"ORSTR(O0%(),P,1)=HEX(85)THEN9805
9804 M=M+1:STR(E%,M,1)=STR(O0%(),P,1):NEXT P
9805 MAT SEARCH E%,%"GOTO"TOW%:GOSUB ' 0:IF W=0THEN9807:STR(E%,W,4)="THEN"
9806 STR(O0%(),K,P-K)=E%:GOTO 9823:DEFFN G(Z)=Z*INT(4/Z)
9807 G0%="THEN":Z=4:GOSUB ' 5(G0%,Z):G0%="PRINT":Z=5:GOSUB ' 5(G0%,Z)
9808 G0%="GOSUB":Z=5:GOSUB ' 5(G0%,Z):G0%="GOSUB'":Z=7:GOSUB ' 5(G0%,Z)
9809 PRINT ,"ИНТЕРПРЕТАТОР ВСТРЕТИЛ НЕДОПУСТИМЫЙ ОПЕРАТОР (IF)":GOTO 9992
9810 RETURN :DEFFN ' 1(N,T,W):GOSUB ' 64(N+1):GOSUB ' 4
9811 STR(O0%(),T+1,4)=N%:STR(O0%(),T+6,W-1)=E%:GOSUB ' 64(N+1)
9812 STR(O0%(),T+W+4,4)="THEN":STR(O0%(),T+W+8,4)=N%
9813 STR(O0%(),T+W+12,1)=":":STR(O0%(),T+W+13,4)="GOTO"
9814 GOSUB ' 64(N+1):U0=LEN(E%):STR(O0%(),T+W+17,4)=N%
9815 STR(O0%(),T+W+21,1)=HEX(85):T=T+W+21:RETURN :DEFFN ' 2(N,T,W0)
9816 GOSUB ' 64(N-1):U0=LEN(E%):STR(O0%(),T+1,4)=N%
9817 STR(O0%(),T+6)=STR(E%,W0,U0-W0+1):STR(O0%(),T+U0-W0+8,1)=HEX(85)
9818 N=N+1:T=T+U0-W0+8:RETURN :DEFFN ' 3(N,T):GOSUB ' 64(N)
9819 STR(O0%(),T+1,4)=N%:STR(O0%(),T+6,6)="RETURN"
9820 STR(O0%(),T+12,1)=HEX(85):T=T+12:RETURN
9821 DEFFN ' 4:INIT (HEX(20))STR(O0%(),K,P-K)
9822 STR(O0%(),K,6)="GOSUB":STR(O0%(),K+6,4)=N%:RETURN
9823 RETURN :DEFFN ' 64(N):INIT (00)N%:CONVERT NTON%,(0000):RETURN
9824 DEFFN ' 5(G0%,Z):MAT SEARCH E%,%STR(G0%,1,Z)TOW%:GOSUB ' 0:GOSUB 9963
9825 IF W=0THEN9827:GOSUB ' 1(N,T,W):W0=W+FNG(Z):GOSUB ' 2(N,T,W0)
9826 GOSUB ' 3(N,T):GOSUB ' 64(N-2):GOSUB ' 4:GOSUB ' 64(N+2):GOTO 9792
9827 RETURN :DEFFN ' 200(K):INIT (HEX(20))E%:M=0:FOR P=KTOK+253
9828 IF STR(O0%(),P,1)="%"ORSTR(O0%(),P,1)=HEX(85)THEN9830
9829 M=M+1:STR(E%,M,1)=STR(O0%(),P,1):NEXT P
9830 U0=LEN(E%):RETURN :DEFFN ' 43(K):GOSUB ' 200(K):GOSUB ' 9
9831 STR(O0%(),K,7)="GOSUB'0":GOSUB ' 201(E%)
9832 STR(O0%(),K+7,K2-K1+1)=STR(E%,K1):N=N+1:GOSUB ' 64(N)
9833 STR(O0%(),T+1,4)=N%:STR(O0%(),T+6,7)="DEFFN'0":L=6
9834 STR(O0%(),T+13)=STR(E%,K1,K2-K1+1):STR(O0%(),T+K2-K1+14)=":"
9835 STR(O0%(),T+K2-K1+15)="Y=X/P-INT(X/P):Y=ROUND(Y#P,0):RETURN"
9836 STR(O0%(),T+K2-K1+51)=HEX(85):T=T+K2-K1+51:GOSUB ' 6(STR(E%,1,L),K):RETURN
9837 DEFFN ' 201(E%):FOR J=1TOLEN(E%):IF STR(E%,J,1)()"("THEN9838:K1=J
```

```
9838 IF STR(E%,J,1)()")"THEN9839:K2=J:GOTO 9840
9839 NEXT J:% ВЫБОРКА ГРАНИЦ ФОРМАЛЬНЫХ ПАРАМЕТРОВ ОПЕРАТОРА
9840 RETURN :DEFFN ' 44(K):GOSUB ' 200(K):GOSUB ' 9:STR(D0%(),K, )="GOSUB ' 1"
9841 GOSUB ' 201(E%):L=8:STR(D0%(),K+9,K2-K1+1)=STR(E%,K1):N=N+1:GOSUB ' 64(N)
9842 STR(D0%(),T+1,4)=N%:STR(D0%(),T+6,7)="DEFFN'1"
9843 STR(D0%(),T+13)=STR(E%,K1,K2-K1+1):STR(D0%(),T+K2-K1+14)=":"
9844 Z=T+K2-K1+14:STR(D0%(),Z+1)="FOR J0=1TOLEN(I%):IF STR(I%,J0,1)()A%THEN"
9845 N=N+1:GOSUB ' 64(N):STR(D0%(),Z+44)=N%:STR(D0%(),Z+48)=":K1=J0"

9846 STR(D0%(),Z+55)=HEX(85):STR(D0%(),Z+56)=N%
9847 STR(D0%(),Z+60)="IF STR(I%,J0,1)()B%THEN":N=N+1:GOSUB ' 64(N)
9848 STR(D0%(),Z+83)=N%:STR(D0%(),Z+87)=":K2=J0:GOTO":GOSUB ' 64(N+1):STR(D0%(),Z+98)=N%
9849 GOSUB ' 64(N-1):STR(D0%(),Z+102)=HEX(85):STR(D0%(),Z+103)=N%:STR(D0%(),Z+107)="NEXTJ0"
9850 GOSUB ' 64(N+1):STR(D0%(),Z+114)=HEX(85):STR(D0%(),Z+115)=N%:STR(D0%(),Z+119)="RETURN"
9851 STR(D0%(),Z+125)=HEX(85):T=Z+125:GOSUB ' 5(STR(E%,1,L),K):RETURN
9852 DEFFN ' 6(E%,K):FOR J=K+1TO10-U0:% ОБРАБОТКА ОПЕРАТОРОВ-ФУНКЦИЙ
9853 IF STR(D0%(),J,L)()STR(E%,1,L)THEN9855:INIT (" ")STR(D0%(),J,K2)
9854 STR(D0%(),J,L+K2-K1+2)=STR(D0%(),K,L+K2-K1+2)
9855 NEXT J:RETURN :DEFFN ' 45(K):GOSUB ' 200(K):INIT (" ")STR(D0%(),K,U0)
9856 STR(D0%(),K,6)="GOSUB":GOSUB ' 201(E%):GOSUB ' 64(N+1):STR(D0%(),K+6,4)=N%
9857 STR(D0%(),T+1,4)=N%:STR(D0%(),T+6)="F":STR(D0%(),T+8)=STR(E%,K2+1,U0-K2):Z=T+U0-K2+8
9858 STR(D0%(),Z)="RETURN":STR(D0%(),Z+7)=HEX(85):T=Z+7:RETURN :DEFFN ' 46(K)
9859 GOSUB ' 200(K):INIT (" ")STR(D0%(),K,U0):A%="[":B%="]":K0=0:GOSUB 9963
9860 GOSUB 9922:STR(D0%(),K,U0)=STR(E%,2,K1-2):STR(D0%(),K+K1,6)="GOSUB":K0=K+K1+6
9861 IF K1=0THEN9864:GOSUB ' 64(N+1):STR(D0%(),K0,4)=N%:STR(D0%(),K0+4,1)=",":GOSUB ' 8
9862 STR(D0%(),T+5)=STR(E%,K1+1,K2-K1-1):STR(D0%(),T+K2-K1+4)=":RETURN":T=T+K2-K1+11
9863 STR(D0%(),T)=HEX(85):STR(E%,K1,1),STR(E%,K2,1)=HEX(20):K0=K0+5:GOSUB 9922:GOTO 9861
9864 STR(D0%(),K0-1,1)=HEX(20):RETURN :DEFFN ' 47(K):GOSUB ' 200(K):GOSUB ' 201(E%)
9865 INIT (" ")STR(D0%(),K,U0):STR(D0%(),K,12)="GOSUB'200(R)":N=N+1
9866 GOSUB ' 64(N):STR(D0%(),T+1)=N%:L=8:GOSUB ' 6(STR(E%,1,L),K)
9867 STR(D0%(),T+1)=N%:STR(D0%(),T+5)="DEFFN'200(R):M,N=D0(1):MATREDIM00(R):FOR"
9868 STR(D0%(),T+45)="J0=1TOR:IFD0(J0))=NTHEN":GOSUB ' 64(N+1):STR(D0%(),T+68)=N%
9869 STR(D0%(),T+72)=":N=D0(J0)":STR(D0%(),T+81)=HEX(85):STR(D0%(),T+82)=N%
9870 STR(D0%(),T+86)="IFD0(J0)(=MTHEN":GOSUB ' 64(N+1):STR(D0%(),T+101)=N%
9871 STR(D0%(),T+105)=":M=D0(J0)":STR(D0%(),T+114)=HEX(85):STR(D0%(),T+115)=N%
9872 STR(D0%(),T+119)="NEXTJ0:RETURN":STR(D0%(),T+132)=HEX(85):T=T+132:RETURN
9873 DEFFN ' 48(K):GOSUB ' 200(K):INIT (" ")STR(D0%(),K,U0):A%="(":B%=")":GOSUB 9922
9874 STR(D0%(),K,17)="GOSUB'203(D,D%,T)":L=8:GOSUB ' 6(STR(E%,1,L),K)
9875 GOSUB ' 64(N+1):STR(D0%(),T+1)=N%:STR(D0%(),T+5)="DEFFN'203(D,D%,T):GOSUB'206(D%)"
9876 STR(D0%(),T+36)=":IFD=1THEN":GOSUB ' 64(N+1):STR(D0%(),T+46)=N%
9877 STR(D0%(),T+50)=":DATALOADBAT(T)A%():GOTO":GOSUB ' 64(N+1)
9878 STR(D0%(),T+74)=N%:GOSUB ' 64(N-1):STR(D0%(),T+78)=HEX(85)
9879 STR(D0%(),T+79)=N%:STR(D0%(),T+83)="DATASAVEBAT(T)A%()":STR(D0%(),T+101)=HEX(85)
9880 GOSUB ' 64(N+1):STR(D0%(),T+102)=N%:STR(D0%(),T+106)="RETURN"
9881 STR(D0%(),T+112)=HEX(85):T=T+112:RETURN :DEFFN ' 49(K):GOSUB ' 200(K):GOSUB ' 9
9882 STR(D0%(),K,13)="GOSUB'201(A%)":GOSUB ' 64(N+1):STR(D0%(),T+1)=N%:GOSUB ' 201(E%)
9883 L=8:GOSUB ' 6(STR(E%,1,L),K):STR(D0%(),T+5)="DEFFN'201(A%):N=1:GOSUB"
9884 GOSUB ' 64(N+1):STR(D0%(),T+28)=N%:STR(D0%(),T+32)=":B0=L0:C0=P0:N=2:GOSUB"
9885 STR(D0%(),T+54)=N%:STR(D0%(),T+58)=":D0=L0:E0=P0:N=B0#16^3+C0#16^2+D0#16+E0:RETURN"
9886 STR(D0%(),T+104)=HEX(85):STR(D0%(),T+108)=N%

9887 STR(D0%(),T+112)="N=VAL(STR(A%,N,1)):L0=INT(N/16):P0=ROUND((N/16-L0)#16,0):RETURN"
9888 STR(D0%(),T+175)=HEX(85):T=T+175:RETURN :DEFFN ' 9:INIT (' ')STR(D0%(),K,U0):RETURN
9889 DEFFN ' 50(K):GOSUB ' 200(K):GOSUB ' 9:STR(D0%(),K,6)="GOSUB"
9890 GOSUB ' 64(N+1):STR(D0%(),T+1)=N%:STR(D0%(),K+6,4)=N%
9891 STR(D0%(),T+6)="S,G0=0:P0=0.1":STR(D0%(),T+19)=HEX(85)
9892 GOSUB ' 64(N+1):M%=N%:GOSUB ' 64(N+1):STR(D0%(),T+20)=M%
9893 STR(D0%(),T+24)="FORX=A TOB-P0STEPP0:S=S+(":GOSUB ' 201(E%)
9894 STR(D0%(),T+48)=STR(E%,K2+3,U0-K2-2):T=T+U0-K2+46
9895 STR(D0%(),T)="#P0:NEXTX:IFABS(S-G0)(=ETHEN":STR(D0%(),T+29)=N%
9896 STR(D0%(),T+33)=":G0=S:P0=P0/10:S=0:GOTO":STR(D0%(),T+56)=M%
9897 STR(D0%(),T+60)=HEX(85):STR(D0%(),T+61)=N%:STR(D0%(),T+65)="RETURN"
9898 STR(D0%(),T+71)=HEX(85):T=T+71:RETURN :DEFFN ' 51(K):GOSUB ' 200(K):GOSUB ' 9:GOSUB ' 201(E%)
9899 STR(D0%(),K,14)="GOSUB'204(K,N)":GOSUB ' 64(N+1):STR(D0%(),T+1)=N%:L=8:GOSUB ' 6(STR(E%,1,L),K)
9900 STR(D0%(),T+5)="DEFFN'204(K,N):C=1:FORJ0=0TOK-1:C=C#(N-J0)/(J0+1):NEXTJ0:C=ROUND(C,0):RETURN"
9901 STR(D0%(),T+81)=HEX(85):T=T+81:RETURN :DEFFN ' 52(K):GOSUB ' 200(K)
9902 GOSUB ' 9:STR(D0%(),K,6)="GOSUB":GOSUB ' 64(N+1):STR(D0%(),K+6,4),STR(D0%(),T+1)=N%
9903 STR(D0%(),T+5)="% %DAEXCH(D,D%,T):GOSUB'206(D%):":STR(D0%(),T+37)="IFD=1THEN"
9904 GOSUB ' 64(N+1):STR(D0%(),T+46)=N%:M%=N%:STR(D0%(),T+50)=":DATALOADDAT(T,S)"
9905 STR(D0%(),T+67)=STR(E%,19,U0-22):GOSUB ' 64(N+1):T=T+U0+45
9906 STR(D0%(),T)=":GOTO":STR(D0%(),T+5)=M%:STR(D0%(),T+9)=HEX(85)
9907 STR(D0%(),T+10)=N%:STR(D0%(),T+14)="DATASAVEDAT(T,S)"
9908 STR(D0%(),T+30)=STR(E%,19,U0-22):STR(D0%(),T+U0+8)=HEX(85):STR(D0%(),T+U0+9)=N%
9909 STR(D0%(),T+U0+13)="RETURN":STR(D0%(),T+U0+19)=HEX(85):T=T+U0+19:RETURN
9910 DEFFN ' 53(K):GOSUB ' 200(K):INIT (' ')STR(D0%(),K,U0):STR(D0%(),K,13)="GOSUB'206(D%)"
9911 GOSUB ' 64(N+1):STR(D0%(),T+1)=N%:L=8:GOSUB ' 6(STR(E%,1,L),K)
9912 STR(D0%(),T+5)="DEFFN'206(D%):IFD%='1CR'THEN":GOSUB ' 64(N+1):STR(D0%(),T+33)=N%
9913 STR(D0%(),T+37)=":IFD%='1CF'THEN":GOSUB ' 64(N+1):STR(D0%(),T+52)=N%
9914 STR(D0%(),T+56)=":IFD%='18R'THEN":GOSUB ' 64(N+1):STR(D0%(),T+71)=N%
9915 STR(D0%(),T+75)=":SELECTDISK18F:GOTO":GOSUB ' 64(N+1):M%=N%:STR(D0%(),T+94)=N%
9916 STR(D0%(),T+98)=HEX(85):GOSUB ' 64(N-3):STR(D0%(),T+99)=N%
9917 STR(D0%(),T+103)="SELECTDISK1CR:GOTO":STR(D0%(),T+121)=M%:STR(D0%(),T+125)=HEX(85)
```

```
9918 GOSUB ' 64(N+1):STR(D0%(),T+126)=N%:STR(D0%(),T+130)="SELECTDISK1CF:GOTO"
9919 STR(D0%(),T+148)=N%:STR(D0%(),T+152)=HEX(85):GOSUB ' 64(N+1):STR(D0%(),T+153)=N%
9920 GOSUB ' 64(N+1):STR(D0%(),T+157)="SELECTDISK18R":STR(D0%(),T+170)=HEX(85)
9921 STR(D0%(),T+171)=N%:STR(D0%(),T+175)="RETURN":STR(D0%(),T+181)=HEX(85):T=T+181:RETURN
9922 DEFFN ' 7(A%,B%,E%):K1,K2=0:FOR J=1TOLEN(E%)
9923 IF STR(E%,J,1)()A%THEN9924:K1=J:GOTO 9925
9924 IF STR(E%,J,1)()B%THEN9925:K2=J:GOTO 9926
9925 NEXT J:% (INTERPRETER) WAS CREATED IN JANUARY 1987 # SKB
9926 RETURN :DEFFN ' 54(K):GOSUB ' 200(K):INIT (' '):STR(D0%(),K,U0):A%="[":B%="]"
9927 GOSUB ' 7(A%,B%,E%):GOSUB ' 64(N+1):STR(D0%(),K,5)="GOSUB":STR(D0%(),K+5,4)=N%

9928 STR(D0%(),T+1)=N%:M%=N%:STR(D0%(),T+5)="FOR%=ATOBSTEP%:IFABS("
9929 STR(D0%(),T+26)=STR(E%,K1+1,K2-K1-1):T=T+K2-K1+25:STR(D0%(),T)=")(=ETHEN"
9930 GOSUB ' 64(N+1):STR(D0%(),T+8)=N%:STR(D0%(),T+12)="'NEXT%:S=S/10:GOTO"
9931 STR(D0%(),T+30)=M%:STR(D0%(),T+34)=HEX(85):STR(D0%(),T+38)=N%
9932 STR(D0%(),T+42)="RETURN":STR(D0%(),T+48)=HEX(85):T=T+48:RETURN
9933 DEFFN ' 55(K):GOSUB ' 200(K):GOSUB ' 201(E%):GOSUB ' 9
9934 STR(D0%(),K,18)="GOSUB'2(N,A%,B%,M)":GOSUB ' 64(N+1):STR(D0%(),T+1)=N%
9935 STR(D0%(),T+5)="DEFFN'2(N,A%,B%,M):DIMA0%253,F0%21,C0%26:F0%="+STR(D0%(),T+50)+HEX(22)
9936 STR(D0%(),T+51)="HHHHSAVEA0%XXX,GGGG":STR(D0%(),T+71)=HEX(22)
9937 STR(D0%(),T+72)=HEX(85):GOSUB ' 64(N+1):STR(D0%(),T+73)=N%
9938 STR(D0%(),T+77)="C0%=":STR(D0%(),T+81)=HEX(22)
9939 STR(D0%(),T+82)="NNNNLOADA0%TTTT,EEEE,KKKK":STR(D0%(),T+107)=HEX(223A)
9940 STR(D0%(),T+109)="STR(F0%,21),STR(C0%,26)=HEX(85)":STR(D0%(),T+140)=HEX(85)
9941 GOSUB ' 64(N+1):STR(D0%(),T+141)=N%:GOSUB ' 64(N+2):M%=N%
9942 STR(D0%(),T+145)="CONVERTNTON0%,(0000):CONVERTNTOM0%,(0000):STR(F0%,1,4)='"
9943 STR(D0%(),T+201)=N%:STR(D0%(),T+205)="'":STR(D0%(),T+206)=HEX(85):GOSUB ' 64(N-1)
9944 STR(D0%(),T+207)=N%:% DYNAMIC PROGRAMM'S MODIFICATION BY (LOAD)
9945 STR(D0%(),T+211)="STR(F0%,12,4),STR(F0%,17,4)=N0%:STR(F0%,16,1)=',':LOADF0%"
9946 STR(D0%(),T+268)=M%:STR(D0%(),T+272)=",":STR(D0%(),T+273)=M%:STR(D0%(),T+277)=","
9947 STR(D0%(),T+278)=M%:STR(D0%(),T+282)=HEX(85):GOSUB ' 64(N+2):STR(D0%(),T+283)=M%
9948 GOSUB ' 64(N+1):STR(D0%(),T+288)="REPLACEL0,A0%,A%,B%:%ЗАМЕНА:"
9949 STR(D0%(),T+316)="STR(C0%,12,4),STR(C0%,17,4)=N0%:STR(C0%,22,4)=M0%:STR(C0%,1,4)='"
9950 STR(D0%(),T+380)=N%:STR(D0%(),T+384)="'":STR(D0%(),T+385)="LOADC0%"
9951 STR(D0%(),T+393)=N%:STR(D0%(),T+397)=",":STR(D0%(),T+398)=N%:STR(D0%(),T+402)=","
9952 STR(D0%(),T+403)=N%:STR(D0%(),T+407)=HEX(85):GOSUB ' 64(N+1)
9953 STR(D0%(),T+408)=N%:STR(D0%(),T+412)="RETURN":STR(D0%(),T+418)=HEX(86)
9954 T=T+418:RETURN :DEFFN ' 56(K):GOSUB ' 200(K):GOSUB ' 201(E%):GOSUB ' 9
9955 STR(D0%(),K,16)="GOSUB'207(D%,A%)":GOSUB ' 64(N+1):STR(D0%(),T+1)=N%
9956 STR(D0%(),T+5)="DEFFN'207(D%,A%):P=0:IFSTR(D%,1,2)()'1C'THEN":GOSUB  64(N+1)
9957 STR(D0%(),T+49)=N%:STR(D0%(),T+53)=":P=8791":STR(D0%(),T+60)=HEX(85)
9958 STR(D0%(),T+61)=N%:STR(D0%(),T+65)="DATALOADBAT(1000+P)B0%:P=0:"
9959 STR(D0%(),T+92)="IFSTR(B0%,1,8)()A%THEN":GOSUB ' 64(N+1)
9960 STR(D0%(),T+114)=N%:STR(D0%(),T+118)=":P=1":STR(D0%(),T+122)=HEX(85)
9961 STR(D0%(),T+123)=N%:STR(D0%(),T+127)="RETURN":STR(D0%(),T+133)=HEX(85)
9962 T=T+133:RETURN :DEFFN ' 87:RETURN CLEAR ALL:K9=K+U0:GOTO 9773:RETURN
9963 REPLACE 09,E%,"&",":":RETURN :DEFFN ' 10:PRINT HEX(030712):PRINT AT(5,20)
9964 PRINT "ПРОГРАММА - ИНТЕРПРЕТАТОР":PRINT :PRINT :PRINT
9965 PRINT "(АВТОР: АЛАДЬЕВ В.Э. # СКБ МПСМ ЗССР ЯНВАРЬ 1987)"
9966 PRINT "0000000000000000000000000000000000000000000000000":RETURN
9967 DEFFN ' 60(K):GOSUB ' 200(K):GOSUB ' 201(E%):GOSUB ' 9:L=8
9968 STR(D0%(),K,19)="GOSUB'202(K,N,A%,O)":GOSUB ' 6(STR(E%,1,L),K):GOSUB ' 64(N+1)

9969 STR(D0%(),T+1)=N%:STR(D0%(),T+5)="DEFFN'202(K,N,A%,O):"
9970 STR(D0%(),T+25)="B=0:DIM(N0%)=2^(8-K):M0%=N0%:AND(STR(N0%,1,1),STR(A%,N,1))"
9971 STR(D0%(),T+83)=":IFSTR(N0%,1,1)=HEX(00)THEN":GOSUB ' 64(N+1):STR(D0%(),T+110)=N%
9972 STR(D0%(),T+114)=":B=1":STR(D0%(),T+118)=HEX(85):STR(D0%(),T+119)=N%
9973 STR(D0%(),T+123)="ONDGOTO":GOSUB ' 64(N+1):STR(D0%(),T+130)=N%
9974 STR(D0%(),T+134)=":BOOL6(STR(A%,N,1),M0%)":STR(D0%(),T+157)=HEX(85)
9975 STR(D0%(),T+158)=N%:STR(D0%(),T+162)="RETURN":STR(D0%(),T+168)=HEX(85):T=T+168:RETURN
9976 DEFFN ' 61(K):GOSUB ' 200(K):GOSUB ' 201(E%):GOSUB ' 9:STR(D0%(),K,15)="GOSUB'200(A%,P)"
9977 GOSUB ' 64(N+1):A%="[":B%="]":GOSUB ' 7(A%,B%,E%):STR(D0%(),T+1)=N%
9978 STR(D0%(),T+5)="DEFFN'200(A%,P):DATA":STR(D0%(),T+25)=STR(E%,K1+1,K2-K1-1)
9979 T=T+K2-K1+25:STR(D0%(),T)=":FORJ0=1TOLEN(A%):FORK0=1TOP:RESTOREK0:"
9980 STR(D0%(),T+39)="READX0:IFJ0()X0THEN":GOSUB ' 64(N+2):STR(D0%(),T+58)=N%
9981 STR(D0%(),T+62)=HEX(85):GOSUB ' 64(N-1):STR(D0%(),T+67)=N%
9982 STR(D0%(),T+71)="P0%=HEX(40):T0%=HEX(50):K0%=HEX(E0):Z0%=HEX(F0)"
9983 STR(D0%(),T+119)=":G0%=STR(A%,J0,1):BOOL6(P0%,G0%):BOOL6(T0%,G0%):BOOL6"
9984 STR(D0%(),T+172)="(K0%,G0%):BOOL6(Z0%,G0%):IFVAL(P0%)>15ANDVAL(T0%)>15"
9985 STR(D0%(),T+224)="ANDVAL(K0%)>15ANDVAL(Z0%)>15THEN":GOSUB ' 64(N+2)
9986 STR(D0%(),T+256)=N%:STR(D0%(),T+260)=":BOOL6(STR(A%,J0,1),HEX(20)):GOTO"
9987 STR(D0%(),T+293)=N%:STR(D0%(),T+297)=HEX(85):GOSUB ' 64(N-1):STR(D0%(),T+298)=N%
9988 STR(D0%(),T+302)="NEXTK0":STR(D0%(),T+308)=HEX(85):GOSUB ' 64(N+1):STR(D0%(),T+309)=N%
9989 STR(D0%(),T+313)="NEXTJ0:RETURN":STR(D0%(),T+326)=HEX(85):T=T+326:RETURN
9990 DEFFN ' 8:STR(D0%(),T+1)=N%:RETURN
9991 PRINT HEX(030712):PRINT "ERROR: "N%;" IN STRING "M%
9992 PRINT "REORGANIZATION OF YOUR INTERPRETED PROGRAMM IS NEEDED":STOP
9993 DIM X0%18:PRINT HEX(0312):CLEAR P9764,9992:ON P60TO9994,9995:RUN
9994 PRINT "ВКЛЮЧИТЬ ПРИНТЕР":STOP "'CONTINUE'":LIST /0C,,9992:GOTO 9999
9995 INPUT "ЗАДАТЬ ДИСК И ИМЯ ПРОГРАММЫ",N%,A%:X0%="9997SELECTDISKVVV"
9996 STR(X0%,18,1)=HEX(85):STR(X0%,15,3)=N%:LOAD X0%9997,9997,9997
9998 SAVE DC TA%,9992:PRINT "ПРОГРАММА "A%;" ЗАПИСАНА НА "N%:GOTO 9999
9999 INPUT "ВАШИ ДЕЙСТВИЯ: ЛИСТИНГ(1),НА ДИСК(2),ЗАПУСК(3)",P0:GOTO 9993
```

языка реализовать интересные программные средства интерпретирующего характера.

Простейший способ расширения возможностей программы "Интерпретатор" состоит во включении в нее нового текста по обработке новых дополнительных операторов БЕЙСИКа. При этом увеличатся объем загрузочного модуля и время интерпретации исходных программ. Существенного расширения возможностей программы "Интерпретатор" можно добиться при использовании средств языка Ассемблер-226, которые дают возможность работать с регистрами ЭВМ и содержимым ОП, отводимый для программы пользователя. Такой подход позволяет ввести ряд дополнительных операторов этого языка для обработки их по программе "Интерпретатор".

В настоящее время программа "Интерпретатор" дополнена двумя новыми режимами. Второй режим предназначен для перевода БЕЙСИК-программ, разработанных для микро-ЭВМ ДВК и ЭЛЕКТРОНИКА-60, в БЕЙСИК-программы, используемые в операционной среде ПК ИСКРА-226. Данный режим интерпретации обеспечивает модуль "А-ИНТЕР" пакета. Модуль"А-ИНТЕР" позволяет не только решать вопросы перевода программного обеспечения с микро-ЭВМ ЭЛЕКТРОНИКА-60 и ДВК на ПК ИСКРА-226, но также расширяет инструментальные средства языка БЕЙСИК указанных микро-ЭВМ относительно операционной среды ПК ИСКРА-226.

Перевод программ с ИСКРА-226 на микро-ЭВМ ДВК и ЭЛЕКТРОНИКА-60 сложнее ввиду значительно более мощных инструментальных средств языка БЕЙСИК ИСКРА-226. Продемонстрируем это на примере реализации оператора SAVE, выгружающего программу из памяти в символьную переменную. Аналогичных средств у диалекта языка БЕЙСИК для ДВК и ЭЛЕКТРОНИКА-60 нет, поэтому алгоритм интерпретации оператора SAVE входной программы может быть следующим: выдается сообщение оператору о необходимости ввести команду SAVE, переписывающую программу из памяти в файл на диске; созданный файл программно преобразуется к требуемому виду и затем по оператору LINPUT считывается в символьную переменную. В этом случае в выходной программе мы вынуждены выйти за рамки собственно языка БЕЙСИК ДВК и использовать команды его интерпретатора. Подобная картина имеет место и с интерпретацией целого ряда других операторов языка БЕЙСИК ИСКРА-226. Следовательно, перевод БЕЙСИК-программ с ПК ИСКРА-226 на микро-ЭВМ ДВК и ЭЛЕКТРОНИКА-60 путем программной интерпретации не представляется, вообще говоря, целесообразным.

Перед изложением собственно второго режима интерпретации необходимо сделать несколько пояснений. Несмотря на значительно более развитые инструментальные средства

языка БЕЙСИК ИСКРА-226 относительно диалектов этого языка для микро-ЭВМ ДВК и ЭЛЕКТРОНИКА-60, оказалось довольно затруднительным (а в некоторых случаях и невозможным) осуществить строго эквивалентное погружение в среду БЕЙСИК ПК ИСКРА-226 некоторых инструментальных средств указанных диалектов языка. Поэтому нами был реализован принцип функциональной эквивалентности операторов входной и выходной программ с учетом последующего выполнения последней в операционной среде БЕЙСИК ПК ИСКРА-226. Такой подход не только упростил задачу интерпретации входной программы, но и породил ряд проблем.

Так, модуль "А-ИНТЕР" интерпретирует входную программу в определенной мере формально, не учитывая в полной мере тонкостей реализации одинаковых по смыслу операторов в разных операционных средах языка БЕЙСИК. Это, в первую очередь, связано с тем, что реализации диалектов языка БЕЙСИК по ряду показателей имеют существенные различия, которые весьма сложно учесть не только при интерпретации входной программы, но и ее последующем выполнении в операционной среде ПК ИСКРА-226. В связи с этим в ряде случаев требуется дополнительная доотладка выходных программ в операционной среде БЕЙСИК ИСКРА-226. Несмотря на это, пакет"ИНТЕРПРЕТАТОР" позволяет существенно упростить и автоматизировать перевод программного обеспечения с микро-ЭВМ ДВК и ЭЛЕКТРОНИКА-60 на ПК ИСКРА-226.

Перечислим теперь основные требования, которым должна удовлетворять входная БЕЙСИК-программа. Прежде всего, входная программа кодируется в полном соответствии с требованиями входного языка БЕЙСИК для микро-ЭВМ ДВК и ЭЛЕКТРОНИКА-60. Параметры операторов входной программы рекомендуется кодировать по крайней мере через один пробел. Номера строк входной программы должны находиться в интервале 10–9653, при этом присутствие строки с номером 10 обязательно. Если входная программа не удовлетворяет этому требованию, то ее следует обрабатывать "ИНТЕРПРЕТАТОРОМ" по частям. Все файлы, определенные во входной программе, будут распределены относительно жесткого диска с адресом 1CR (тип ЕС 5260–01). В процессе интерпретации в выходной программе могут генерироваться следующие массивы и переменные: 0¤(256)1, 01¤, 02¤, 03¤, 05¤, 04¤253, R9¤253, I0,W0, 00, 01. Поэтому при использовании их во входной программе следует проявлять внимательность. Следующие требования к входной программе являются обязательными:

1. Значения целых переменных должны находиться в пределах от –7999 до +7999;

2. Длина символьной переменной не должна превышать 253 байта;

3. Индексы массивов не должны принимать нулевых значений;

4. Недопустимо использование унарных знаков минуса или плюса;

5. В операторе использование функции RND допускается не более одного раза;

6. Не допускается использование оператора CALL – вызов ассемблеровской программы;

7. В символьном выражении операцию (+/&) можно использовать не более одного раза;

8. Недопустимо считывать по оператору READ числовую константу в символьную переменную;

9. Логические номера каналов ввода/вывода должны быть в интервале 0–7;

10. Нельзя использовать функции символьных аргументов: DATo, CLKo и все функции преобразования символов в код КОИ-7 и наоборот;

11. В операторе ⟨NAME TO⟩ можно указывать только имена файлов, а не логические номера устройств с файлами;

12. Функции пользователя, определенные оператором DEF, обязательно должны в скобках содержать формальные параметры.

Если входная программа не удовлетворяет перечисленным требованиям, то возникает аварийная ситуация либо в процессе ее интерпретации (пункты 6, 9–11), либо выполнения выходной программы (остальные пункты). Обнаружив несоблюдение основных требований, модуль "А-ИНТЕР" информирует об этом пользователя и дает возможность продолжить интерпретацию входной программы. В случае непредусмотренной аварийной ситуации во время интерпретации входной программы следует выдать команду: RUN 9985.

Практическое использование пакета "ИНТЕРПРЕТАТОР" во втором режиме показало, что несмотря на вышеперечисленные ограничения, большинство массовых программ на языке БЕЙСИК ДВК и ЭЛЕКТРОНИКА-60 вполне удовлетворительно переводятся для использования их в операционной среде ИСКРА-226. Остальные программы после интерпретации их пакетом требует относительно простых доработки и доотладки. Экспертные оценки показали высокий уровень автоматизации по переводу БЕЙСИК-программ в операционную среду ИСКРА-226, обеспечиваемый режимом 2 "ИНТЕРПРЕТАТОРА".

Суть интерпретации входной программы в режиме 2 состоит в следующем. Модуль "А-ИНТЕР" располагает шестью полями длиной по 253 байта для работы со строкой и отдельным оператором входной программы:

E1¤ – поле программной строки;

E¤ – поле интерпретируемого оператора;

E2¤ – E5¤ – дополнительные рабочие поля

и тремя одномерными массивами одинаковой длины, определяемой пользователем в самом начале работы пакета:

0₀¤( ) – массив со входной программой;

G₀¤ ( ) – массив с выходной программой без расширения;

S₀¤ ( ) – массив с расширением выходной программы.

После загрузки и запуска модуля "А-БЕЙСИК" пакета и получения запроса на требуемый режим интерпретации необходимо ответить номером заказа "2". После этого с дискового тома, содержащего пакет "ИНТЕРПРЕТАТОР", загружается второй модуль "А-ИНТЕР" и начинается собственно второй режим интерпретации. В течение всего процесса интерпретации на дисплее высвечивается сообщение:

ИДЕТ ИНТЕРПРЕТАЦИЯ ВАШЕЙ ПРОГРАММЫ ДЛЯ ИСПОЛЬЗОВАНИЯ ЕЕ В СРЕДЕ ‹БЕЙСИК 02›!

Сам алгоритм интерпретации в общих чертах сводится к следующему.

Входная программа по оператору SAVE выгружается в массив 0₀¤( ). Определяется длина входной программы и максимальный номер ее программных строк. Затем из массива 0₀¤( ) выбирается первая программная строка и помещается в рабочее поле E1¤. Строка находится здесь весь период, пока не завершится интерпретация последнего оператора строки, предоставляя модулю "А-ИНТЕР" необходимую информацию. Из рабочего поля E1¤ выделяется очередной оператор строки входной программы и помещается в рабочее поле E¤. Для интерпретации оператора могут потребоваться дополнительные рабочие поля E2¤ – E5¤, в которых формируются строки помеченных или обычных подпрограмм. Эти подпрограммы генерируются по соответствующим алгоритмам и моделируют интерпретируемые операторы входной программы. Нумерация строк в такой подпрограмме осуществляется с шагом 1 и все строки после их генерации помещаются в символьный массив S0¤( ) расширений выходной программы, начиная с первого свободного байта. Номер первой строки очередной генерируемой подпрограммы определяется на единицу большим текущего номера строки в массиве S0¤( ). Для возможности обращения к сгенерированной подпрограмме в интерпретируемом операторе (поле E¤) генерируется соответствующий оператор GOSUB' (GOSUB) и, возможно, перед проинтерпретированным оператором E¤ генерируется некоторый набор вспомогательных операторов. В отдельных случаях интерпретация оператора из E¤ не требует ни генерирования программ, ни вспомогательных операторов перед интерпретируемым оператором в E¤.

После завершения формирования поля E¤, содержащего проинтерпретированный оператор, операторы из него заносят-

ся в массив GO¤ ( ) выходной программы без расширения, начиная с первого свободного байта. После завершения интерпретации последнего оператора входной программы расширение из символьного массива SO¤ ( ) переписывается в символьный массив GO¤( ), завершая формирование выходной программы. Выходная программа из массива GO¤( ) переписывается в массив 0o¤ ( ) и пользователь информируется о размерах входной и выходной программ, а также времени интерпретации. Затем выходная программа из массива 0o¤( ) загружается по оператору LOAD в память и пользователю предоставляется возможность: получить листинг, переписать на заданный дисковый том под указанным именем или выполнить в операционной среде БЕЙСИК ИСКРА-226.

Здесь лишь целесообразно упомянуть о тех основных предположениях, которые были использованы при проектировании общего алгоритма интерпретации входных программ в режиме 2.

Вышеперечисленные ограничения на входную программу для данного режима интерпретации были вызваны двумя основными причинами:

– ограничениями инструментальных средств операционной среды БЕЙСИК ИСКРА-226;

– сложностью и громоздкостью алгоритмов интерпретации. В качестве иллюстрации к первой причине можно привести пример по ведению службы времени. В операционной среде БЕЙСИК ИСКРА-226 такая служба, кроме обычного таймера, отсутствует. Тогда как операционная система, скажем, ДВК позволяет языку БЕЙСИК использовать функции даты и времени для получения в программу пользователя текущих дат и времени.

В качестве примера по второй причине можно привести интерпретацию функции ASC(A¤), которая возвращает значение символа A¤ в коде КОИ-7. В этом случае алгоритм моделирования данной функции имеет дело с громоздкой перекодировочной таблицей.

Здесь же мы рассмотрим только основные соглашения, которые были приняты при создании "А-ИНТЕР" и которые вызваны различиями, и порой существенными, между операционными средами ПК ИСКРА-226, микро-ЭВМ и ЭЛЕКТРОНИКА-60.

Для операторов PRINT, PRINTUSING и PRINT #⟨логический номер⟩ USING принят формат вывода информации на дисплей и принтер, используемый в БЕЙСИК ИСКРА-226.

Все операторы ввода/вывода соотносятся с жестким диском 1CR и логические номера устройств должны быть в интервале 0–7. Открытие и закрытие файлов на дисковых томах осуществляется согласно требованиям БЕЙСИК ИСКРА-

226. На признак "конец файла" может проверяться только текущий файл программы пользователя.

Операторы CHAIN и OVERLAY вызова программ в память также интерпретируются с учетом требований БЕЙСИК ИСКРА-226. Ввиду отличий языков Ассемблера для ИСКРА-226 и ДВК оператор CALL вызова ассемблеровских программ запрещен. Для доотладки выходных программ следует руководствоваться требованиями и рекомендациями, имеющимися в технической документации по ПК ИСКРА-226. Остальные принципы интерпретации операторов входной программы для использования ее в операционной среде БЕЙСИК ИСКРА-226 легко усматриваются из вышеприведенного листинга примера входной программы.

В процессе интерпретации входной программы производится обработка возникающих особых ситуаций. При возникновении особой ситуации выдается базовое сообщение (возможны некоторые уточняющие к нему пояснения):

ОШИБКА ИНТЕРПРЕТАЦИИ ОПЕРАТОРА: ⟨оператор⟩ В СТРОКЕ ⟨строка⟩

ИСПРАВИТЬ ВХОДНУЮ ПРОГРАММУ – РАБОТУ ПРОДОЛЖИТЬ (1-Д/2-Н)? идентифицирующее строку входной программы и оператор в ней, вызвавший особую ситуацию. После этого пользователю предоставляется возможность либо продолжить интерпретацию, либо выйти на ее завершение с возможными последующими действиями: получить листинг выходной программы, записать ее на диск, выполнить или вернуться к началу работы пакета "ИНТЕРПРЕТАТОР".

Приведем теперь листинг модуля "А-ИНТЕР", позволяющий в полной мере представить реализацию второго режима интерпретации.

После успешного завершения интерпретации входной программы модуль "А-ИНТЕР" выдает пользователю информацию, аналогичную той, что и модуль "А-БЕЙСИК" первого режима интерпретации. Тут же пользователю предоставляется возможность повторной обработки выходной программы модулем "А-ИНТЕР". Такая ситуация возникает, например, если в операторе ⟨IF THEN⟩ входной программы содержатся другие операторы, требующие интерпретации. При работе с модулем "А-ИНТЕР" в режиме интерпретации набора входных программ обработку, начиная со второй загруженной программы, можно начинать по команде: RUN 9671. После выполнения выходной программы, вышедшей на оператор END, вернуться в "ИНТЕРПРЕТАТОР" можно по команде: RUN9999. В случае непредусмотренной аварийной ситуации в процессе интерпретации следует выдать команду: RUN9985.

Наконец, второй режим интерпретации позволяет БЕЙСИК-

программам ИСКРА-226 использовать два дополнительных инструментальных средства:

операцию символьного сложения (+/&);

функции пользователя широкого назначения.

Для возможности использования этих средств входная программа составляется согласно требованиям БЕЙСИК ИСКРА-226 с той лишь разницей, что разделителем операторов в программной строке является символ "\", а не ":". После загрузки такой программы следует выдать команды: PO=40: RUN 9658 с последующим определением второго режима интерпретации.

На этом завершается обсуждение второго режима интерпретации и мы переходим к рассмотрению третьего режима интерпретации входной программы.

Используя подход и принципы, положенные в основу создания режимов 1 и 2 пакета "ИНТЕРПРЕТАТОР", можно создавать различного назначения программные средства интерпретирующего типа для использования их в операционной среде БЕЙСИК ИСКРА-226. В частности, пользователь может на этой основе создавать различные диалекты языка БЕЙСИК, диалекты ряда других языков программирования, языки специального назначения и так далее.

Здесь описывается последний режим интерпретации пакета "ИНТЕРПРЕТАТОР", обеспечиваемый модулем "В-ИНТЕР". Данный режим позволяет пользователю писать программы на русском диалекте базового языка БЕЙСИК с последующим использованием результатов обработки их модулем "В-ИНТЕР" в операционной среде БЕЙСИК ИСКРА-226. Входная программа на русском диалекте языка загружается в память для обработки модулем "В-ИНТЕР" аналогично случаю ранее описанных режимов интерпретации 1 и 2. Приведем теперь необходимые пояснения по русскому диалекту базового языка БЕЙСИК.

На основе анализа языка БЕЙСИК ИСКРА-226 было выбрано его подмножество операторов, позволяющее создавать практически любые программные средства. Ключевые слова и операторы выбранного подмножества языка получили русскую транскрипцию согласно следующей таблице соответствий, в которой пронумерованным русским ключевым словам и операторам языка поставлены в соответствие английские эквиваленты языка БЕЙСИК ИСКРА-226.

Указанные в таблице ключевые слова и операторы русской транскрипции оформляются во входной программе аналогично их английским эквивалентам в языке БЕЙСИК ИСКРА-226. Из ключевых слов можно образовывать другие допустимые конструкции, например: ПО ОШИБКЕ A¤, B¤ ПЕРЕЙТИ 200 или

1. ПРИ КОНЦЕ ФАЙЛА	IF END THEN
2. УСТАНОВИТЬ	SELECT
3. РАЗМЕР	DIM
4. ПЕЧАТЬ	PRINT
5. ДЛЯ	FOR
6. ШАГА	STEP
7. ОСТАНОВ	STOP
8. ОБЩИЕ	COM
9. УДАЛИТЬ	SCRATCH
10. ПОДПРОГРАММА	DEFFN'
11. ВЫГРУЗИТЬ	SAVE
12. КОНЕЦ	END
13. ПРЕОБРАЗОВАТЬ	CONVERT
14. ЗАМЕНИТЬ	REPLACE
15. ПРОПУСК	DSRIP
16. МАТРВЫВОД	MATPRINT
17. ЛИСТИНГ	LIST
18. МАКЕТ	%
19. КЛЮЧВВОД	R KEYIN
20. ГОТОВИТЬ	SCRATCH
21. КОПИЯ1	MOVE
22. ВВОДКОР	LINPUT
23. ПЕРЕНУМЕРОВАТЬ	RENUMBER
24. ТЕКСТ	REM
25. ВВЕСТИ	INPUT
26. ПЕРЕЙТИ	GOTO
27. ДО	TO
28. ЦИКЛ	NEXT
29. ВОЗВРАТ	RETURN
30. ЗАГРУЗИТЬ	LOAD
31. ВЫПОЛНИТЬ	GOSUB
32. ПО	ON
33. ВЫППРОГ	GOSUB'
34. ОШИБКЕ	ERROR
35. НА	TO
36. ОБЕРНУТЬСЯ	DBACKSPACE
37. СТАРТ	RUN
38. ВВОДМАТР	MATINPUT
39. ВЫВОД*16	HEXPRINT
40. МАКЕТВЫВОД	PRINTUSING
41. ЗАММАССИВ	MATREDIM
42. ДИСК	DISK
43. КОПИЯ2	COPY
44. –"–	–"–

УСТАНОВИТЬ ПЕЧАТЬ ОС (164). В связи со сказанным, дополнительных пояснений по вышеперечисленным ключевым словам и операторам русской транскрипции не требуется. Русский диалект дополнительно перечисленным инструментальным средствам располагает следующими операторами.

Оператор ПУСТЬ имеет следующий формат

ПУСТЬ ⟨список символьных переменных⟩ = ⟨значение⟩ и эквивалентен оператору INIT. Например, оператор ПУСТЬ А¤, В¤ = "А" входной программы эквивалентен оператору INIT ("А") А¤, В¤ выходной программы.

Оператор ЕСЛИ имеет следующий формат

ЕСЛИ ⟨логическое условие⟩ [ПЕРЕЙТИ ⟨номер строки⟩](1) [ТО ⟨список операторов⟩](2)

и является обобщением оператора ⟨IF⟩ условного перехода языка БЕЙСИК ИСКРА-226. Первая форма оператора эквивалентна оператору ⟨IF⟩, тогда как вторая является его существенным расширением. Данная форма позволяет в случае истинности ⟨логического условия⟩ выполнять операторы из ⟨списка операторов⟩. В списке операторов нельзя использовать оператор ПЕРЕЙТИ. Например: ЕСЛИ А = 5 ТО ВЫПОЛНИТЬ 145: ПЕЧАТЬ Х, У:А=19.42.

Оператор ФУНКЦИЯ имеет следующий формат

ФУНКЦИЯ FN⟨имя⟩ ⟨признак⟩(⟨формальные параметры⟩)= ⟨функциональное выражение⟩

и служит для определения функций пользователя широкого назначения. В качестве параметра ⟨имя⟩ можно использовать любую латинскую букву; параметр ⟨признак⟩ принимает одно из двух значений: "%" или "¤" и определяет соответственно целочисленную или символьную функцию, тогда как отсутствие этого признака определяет функцию действительных значений. Наличие ⟨формальных параметров⟩ у функции пользователя обязательно. Обращение к определенной таким образом функции имеет вид: FN ⟨имя⟩ (⟨формальные параметры или их значения⟩). Организация работы с функциями, определенными оператором ФУНКЦИЯ, аналогична использованию функций пользователя языка БЕЙСИК ДВК, рассмотренных в предыдущем разделе.

Оператор ЧИТАТЬ имеет следующие две формы

ЧИТАТЬ ⟨список перемен⟩ [ИЗ ⟨файл⟩ #⟨ФАУ⟩](1) [параметры отсутствуют](2)

и служит для чтения логической записи из файла с заданным параметром ⟨файл⟩ именем на диске с указанным ⟨ФАУ⟩ и присвоения значений переменным из ⟨списка переменных⟩. Вторая форма оператора относится только к текущему файлу, т. е. файлу, к которому было последнее обращение по операторам ПИСАТЬ (первая форма) или ЧИТАТЬ (первая форма).

Оператор ПИСАТЬ имеет также две формы

ПИСАТЬ ⟨список переменных⟩ [В ⟨файл⟩#⟨ФАУ⟩ (1); параметры отсутствуют (2)]

и служит для записи информации в виде единой логической записи в заданный файл на указанном дисковом томе. Оператор в первой форме является обратным оператору ЧИТАТЬ. Оператор во второй форме в качестве параметра ⟨список переменных⟩ может принимать значения: END, CLOSE или список переменных. Во второй форме оператор ПИСАТЬ относится также только к текущему файлу.

При работе с дисковыми операторами ПИСАТЬ и ЧИТАТЬ пользователь освобожден от необходимости открывать файлы на дисках, т. к. эту функцию выполняет за него оператор LIMITS,входящий в состав генерируемых модулем "В-ИНТЕР" подпрограмм выходной программы, моделирующих дисковые операторы. При попытке поместить логическую запись в отсутствующий на диске файл соответствующая подпрограмма выходной программы открывает на указанном диске новый файл с заданным именем и размером в 300 секторов и помещает в него первую запись. Все записи (кроме первой) следует читать или записывать операторами ЧИТАТЬ или ПИСАТЬ второй формы соответственно. В случае возникновения аварийных ситуаций при работе дисковых операторов ЧИТАТЬ или ПИСАТЬ пользователю выдаются соответствующие подробные сообщения.

В качестве разделителя операторов в программной строке русского диалекта языка используется символ "\", а не ":". Правила оформления операторов русского диалекта языка (за исключением операторов ПУСТЬ, ЕСЛИ (вторая форма), ФУНКЦИЯ, ЧИТАТЬ и ПИСАТЬ) полностью соответствуют правилам оформления эквивалентных им операторов языка БЕЙСИК ИСКРА-226 и здесь обсуждаться не будут. Другие основные конструкции языка БЕЙСИК ИСКРА-226 также сохраняют силу для русского диалекта: переменные, константы, массивы, их типы, операции и т. д.

Модуль "В-ИНТЕР" является модификацией модуля "А-ИНТЕР", описанного в предыдущем разделе. Принципиальная схема интерпретации входной программы на русском диалекте базового языка БЕЙСИК полностью соответствует схеме модуля "А-ИНТЕР". Сам же алгоритм интерпретации входной программы значительно проще и в общих чертах сводится к следующему.

Алгоритм выгрузки входной программы в символьный массив 0o¤( ) и выборки на интерпретацию очередного оператора программы совпадает с аналогичной процедурой модуля "А-ИНТЕР". После выборки очередного оператора из массива

0o¤( ) в поле E¤ операторов формируется символьная матрица B¤ (50,2), представляющая собой таблицу соответствий между ключевыми словами и операторами языка БЕЙСИК ИСКРА-226 и их русскими аналогами. Основное содержимое этой матрицы в табличном виде представлено в начале данного раздела. Используя данную матрицу в качестве своего рода перекодировочной таблицы, модуль "B-ИНТЕР" каждое вхождение русского слова в интерпретируемом операторе из поля E¤ заменяет соответствующим ему английским эквивалентом.

Эту часть алгоритма интерпретации реализуют программные строки с номерами 9688–9751. Интерпретация остальных операторов русского диалекта языка осуществляется программными строками согласно следующей таблицы:

Оператор	Номера программных строк модуля "B-ИНТЕР"
ПУСТЬ	9752–9756
ЕСЛИ	9757–9766
ФУНКЦИЯ	9768–9776
ЧИТАТЬ	9778–9820
ПИСАТЬ	9821–9834

Остальная часть листинга модуля "B-ИНТЕР" практически полностью заимствована у модуля "A-ИНТЕР". Из приведенной общей схемы интерпретации входной программы нетрудно заметить, что операторы русского диалекта языка, кроме операторов ПУСТЬ, ЕСЛИ, ФУНКЦИЯ, ЧИТАТЬ и ПИСАТЬ, полностью эквивалентны соответствующим им операторам английской транскрипции языка БЕЙСИК ИСКРА-226. Тогда как указанные пять операторов потребовали дополнительных описания и пояснения. Выдаваемые в процессе интерпретации входной программы модулем "B-ИНТЕР" сообщения аналогичны сообщениям двух вышеописанных режимов пакета "ИНТЕРПРЕТАТОР" и дополнительных пояснений не требуют.

Русский диалект базового языка БЕЙСИК в первую очередь предназначен для быстрого освоения основ программирования на ПК ИСКРА-226. Небольшое число операторов, русская мнемоника и простота работы с файлами данных на дисковых томах позволяют пользователю, не имеющему до этого дела с программированием, весьма быстро получить необходимые навыки по программированию прикладных задач для последующего их использования в операционной среде БЕЙСИК ПК ИСКРА-226.

При подготовке входной программы на русском диалекте языка следует соблюдать следующие простые соглашения. Операторы, определяемые матрицей соответствий B¤ (50,2), кодируются и используются в полном соответствии с их

английскими аналогами языка БЕЙСИК ИСКРА-226. Поэтому при их использовании следует руководствоваться материалами по ИСКРА-226. Это же относится и ко всем другим конструкциям языка: константам, переменным и массивам разных типов, элементарным математическим и символьным функциям, форматам используемых данных, арифметическим операциям и так далее.

Пять операторов русского диалекта языка (ПУСТЬ, ФУНКЦИЯ, ЧИТАТЬ, ПИСАТЬ и ЕСЛИ) должны использоваться в полном соответствии с приведенным выше их описанием. В связи со спецификой использования оператора ФУНКЦИЯ, позволяющего пользователю в своей программе определять функции весьма общего типа, не рекомендуется использовать в программе одновременно оператор ФУНКЦИЯ и операторы ПОДПРОГРАММА с номерами в диапазоне 65–90. Несмотря на высокий уровень автоматизации процедур обмена информацией с дисковыми томами, дисковые операторы следует использовать достаточно внимательно, предварительно хорошо уяснив различие между двумя их формами. Следует помнить, что вторая форма дисковых операторов относится только к текущему файлу и используется, как правило, при циклическом обращении к файлу данных.

В заключение еще раз необходимо подчеркнуть, что описанные здесь пакет "ИНТЕРПРЕТАТОР" и вопросы его использования служат не только целям предоставления пользователю нового программного средства по расширению инструментальных средств языка БЕЙСИК ИСКРА-226 и решению вопросов перевода программного обеспечения с микро-ЭВМ ДВК и ЭЛЕКТРОНИКА-60 на ПК ИСКРА-226. Важная роль настоящей главы состоит также в том, что на примере организации пакета и его листинга читатель имеет возможность ознакомиться с практической реализацией в среде БЕЙСИК ИСКРА-226 достаточно сложных программных систем интерпретирующего типа и получить представление о практике использования большинства основных операторов языка БЕЙСИК при создании целого ряда программных конструкций.

Дальше рассмотрим основные принципы и простейшую методику подготовки и решения задач с помощью ЭВМ "Искра-226" на примере конкретной научной задачи из теории чисел, а также некоторые вопросы работы с данной ЭВМ. При написании программы ставилась цель максимально (для иллюстрации) использовать основные операторы языка БЕЙСИК, что в определенном смысле снизило эффективность самой программы.

5. МЕТОДИКА ПОДГОТОВКИ И РЕШЕНИЯ ЗАДАЧ НА ЭВМ "ИСКРА-226"

5.1. Задача Улама

В целях демонстрации методики подготовки задач для их решения на ЭВМ "Искра-226" выберем хорошо известную из теории чисел задачу, названную С. Уламом "Последовательности однозначно определенных сумм" (ПООС). Теоретическое и экспериментальное исследование решений одной модификации ПООС проводилось в [27, 28]. Использовались ЭВМ "Минск-32", ЕС и СМ ЭВМ. Приведенная ниже программа "ПООС", написанная специально для ЭВМ "Искра-226", позволяет в диалоговом режиме решать эту задачу.

Кратко суть задачи сводится к следующему. Берется множество M= {1,2,3...} целых чисел и на нем определяется бинарная операция A: X + Y→P, где X, Y, P ∈ M. Элементы P образуют множество M'. На операцию A накладываются следующие ограничения: начав с P, равных a и b (a < b), следующие P получаем как суммы каких-либо двух предыдущих элементов, но не включаем в них те суммы, которые можно получить более чем одним способом; при этом сами с собой числа не складываются и в сложении обязательно участвует самый правый элемент уже существующего отрезка (a, b) ПООС.

Близнецами в ПООС будем называть пары элементов, отличающихся между собой по значению на p=p (a, b). В дальнейшем любые множества пар близнецов будем обозначать просто Б (p). Например, Б (a + b) – множество пар близнецов вида p (a, b) = a + b.

В программе "ПООС" заложена возможность исследования:

частичных плотностей ρ_k ПООС, начиная с заданного элемента, и степени изменения плотностей ρ_k ($\Delta\rho_k$);

степени роста значений элементов ПООС, начиная с заданного;

частичных плотностей пар близнецов относительно ПООС;

изменения расстояния между ближайшими парами близнецов;

числа пар близнецов в заданном интервале ПООС, а также исследования перечисленных выше свойств ПООС при произвольных начальных элементах a, b и исследования ПООС1 (a, b). При этом ПООС1 (a, b) отличается от ПООС (a, b) только тем, что не требуется обязательного участия в операции сложения правого крайнего элемента уже сформированного отрезка (a, b). Заметим также, что оба варианта ПООС наряду с самостоятельным интересом в теории чисел имеют ряд биологических интерпретаций.

Программа "ПООС" на языке БЕЙСИК версии ЭВМ "WANG-2200", листинг которой приводится ниже, позволяет пользова-

телю в диалоговом режиме на ЭВМ "Искра-226" успешно исследовать свойства ПООС обеих модификация. Но прежде изложим вкратце методику подготовки и решения задач с помощью данной ЭВМ.

На первом этапе подготавливаются НГМД, которые понадобятся на стадии отладки и решения задачи. Подготовка НГМД сводится к инициализации и разметке их согласно следующей программе (не нарушая общности и там, где это возможно, все пояснения будем относить к НГМД типа R):

```
10 PRINT  /18, HEX (1B000100000400)
20 SCRATCH DISK  RLS =2, END = 1000
30 SELECT  LIST OC : LI ST DCR : END
```

Эта программа позволяет не только проинициализировать НГМД, отвести место на нем под каталог в 1000 секторов (включая два сектора под УК), но и произвести для проверки контрольную распечатку УК на АЦПУ. Данную работу можно выполнить также, используя сервисные средства (например, программу "Нива" [26]).

На втором этапе готовятся файлы (программный и для ввода данных), которые понадобятся для отладки программы с данными. При этом используется программа:

```
10 DATA SAVE DC OPEN R ¤  (300) "DATA"
20 SAVE  DC  R¤T  (22) "ПООС"
30 SELECT  LIST OC :  LIST DCR :  VERIFY R :  END
```

После выполнения этой программы на диске R заводятся файл данных " DATA " (на 300 секторов), программный файл "ПООС" (на 22 сектора), весь диск проверяется и на АЦПУ печатается содержимое УК:

```
REMOVABLE CATALOG
INDEX SECTORS = 00002
END CAT. AREA = 01000
CURRENT END = 00324
NAME     TYPE     START     END     USED
DATA     D        00002     00301   00001
ПООС     P        00302     00323   00021
```

Теперь наступает наиболее важный этап — написание и отладка программы. Успех на этом этапе определяют два основных фактора: знание сущности задачи и возможностей ЭВМ, на которую ориентирована программная реализация задачи. Будем считать, что с этими факторами дело у нас обстоит удовлетворительно. Тогда, составив алгоритм решения задачи (в виде схемы, словесного описания на каком-либо алгоритмическом языке и т. п.), опишем его на языке БЕЙСИК.

При этом писать и отлаживать программу можно частями (вплоть до пооператорного уровня).

Написав программу или ее часть, введем ее с клавиатуры в память ЭВМ и начнем вести отладку. БЕЙСИК позволяет довольно легко отлаживать программы (см. [11]) и постоянно контролировать все вводимые в них изменения. При этом следует (если он есть) использовать опыт программирования на других ЭВМ.

Отработав какую-то часть программы, поместим ее на НМД, используя операторы языка БЕЙСИК в режиме НСЧ:

```
: SCRATCH R "ПООС"
: SAVE DC R T("ПООС") "ПООС"
```

В результате на НМД созданный ранее программный файл "ПООС" сначала помечается как ликвидируемый, затем на его место под тем же именем помещается программа (или ее часть), отладка которой только что велась.

Перед очисткой ОП для другой работы (оператор CLEAR) рекомендуется вывести посредством оператора LIST/OC листинг программы на АЦПУ, чтобы иметь возможность продолжить ее отладку за столом. Для того, чтобы следующий раз продолжить работу по отладке, диск с программой "ПООС" следует поставить на НГМД и загрузить ее в память ЭВМ:

```
: LOAD DC R "ПООС"
```

Проверить наличие программы в ОП можно посредством оператора LIST или LIST S (если нужно "пролистать" ее на дисплее по страницам, которые содержат по 23 строки программы каждая).

С вызванной программой можно продолжить работу по ее доотладке. В распоряжении пользователя для этого имеется достаточное количество средств. Имея почти 20-летний опыт работы с устройствами ВТ разных классов и типов, с полной уверенностью можем заявить, что по средствам отладки программ ЭВМ "Искра-226" во многих отношениях превосходит предыдущие такие средства ВТ, хотя в ряде отношений и уступает им.

В первую очередь это касается процесса выполнения программ. ЭВМ "Искра-226" оснащена относительно слабо развитыми средствами диагностики аварийных ситуаций (80 типов) и средствами восстановления работоспособности. Поэтому пользователю рекомендуется предусмотреть в своей программе (используя оператор ON ERROR) обработку основных возможных аварийных ситуаций, примером чего может служить программа "ПООС" (см. ее листинг).

Для целей отладки пользователь может включать в текст программы временные операторы, которые после ее отладки

исключаются. Отладку программы следует вести по всем ее частям, что несложно, поскольку БЕЙСИК позволяет начать работу программы с любой ее строки: RUN XXXX (XXXX-НС). К сожалению, среди программ, имеющихся на сегодня для ЭВМ "Искра-226", довольно часто встречаются еще не до конца отлаженные.

Следует помнить, что программа составляется один раз и продумать ее нужно так, чтобы она была по возможности проще в эксплуатации, устойчивой к возникающим аварийным ситуациям и адаптируемой к основным возможным расширениям.

Убедившись в том, что отладка проведена полностью и программа выдает нужные результаты, окончательный ее текст следует поместить на НМД. При этом рекомендуется сделать дополнительные копии программы (лучше на разных накопителях) и удалить все файлы ("мусор"), использовавшиеся при ее отладке. Это можно выполнить с помощью операторов SCRATCH и MOVE, а также специальных средств, описанных ниже.

Листинг программы и распечатку УК диска (дисков) с программой и данными также рекомендуется хранить в отдельной папке, куда вносятся все последующие изменения программы и содержимого УК. Там же следует хранить и экземпляр инструкции по работе с программой, если ее разработчиком не предусмотрена возможность выдачи этой инструкции пользователю на АЦПУ и (или) дисплей.

Изложив кратко методику подготовки и решения задач на ЭВМ "Искра-226" перейдем к рассмотрению листинга программы "ПООС", чтобы получить представления о том, как она оформляется и как используются в ней основные операторы языка БЕЙСИК. Примерами использования операторов БЕЙСИКа в сервисных программах приведены ниже.

```
0 PRINT " ВЫ РАБОТАЕТЕ С ПРОГРАММОЙ <ПООС>"
1 PRINT "#####################################"
2 PRINT " (АВТОР: АЛАДЬЕВ Б. - СКБ МПСМ ЗССР "
3 PRINT "#####################################"
4 DIM A(3000):PRINT :PRINT
5 INPUT "РАБОТАТЬ С ПООС(1,2)(1),ПООС(A,B)(2),ПООС1(A,B)(3)",L
7 INPUT "ВВЕСТИ ПЕРВЫЕ 5 ЧЛЕНОВ ПООС",A1,A2,A3,A4,A5
8 A(1)=A1:A(2)=A2:A(3)=A3:A(4)=A4:A(5)=A5
9 PRINT "НАЧАЛО ПООС: ";A1;A2;A3;A4;A5
10 INPUT "РАБОТА С ПООС ВПЕРВЫЕ(1-ДА/2-НЕТ)",L:ON L GOTO15:GOTO 31
11 PRINT "РАБОТА С ПРОГРАММОЙ ВПЕРВЫЕ ИЛИ НЕТ ?":DIM A$3
12 INPUT "ОТВЕТ (YES/NO)",A$:IF A$="YES"THEN13:PRINT "ОШИБКА":GOTO 11
13 DIM E%4,R%4,A(3000):A(1)=1:A(2)=2:A(3)=3:A(4)=4:A(5)=6:A(6)=8
14 A(7)=11:A(8)=13:A(9)=16:A(10)=18:A(11)=26:A(12)=28
15 PRINT "#############################################################"
16 PRINT "    НОМЕР     #    ЗНАЧЕНИЕ     #       ПЛОТНОСТЬ"
17 PRINT "    ЧЛЕНА     #    ЧЛЕНА        #       ОТРЕЗКА ПООС "
18 PRINT "#############################################################"
19 FOR I=1TO3000STEP1:IF A(I)=0THEN20:NEXT I
20 I=I-1
21 K=1:N=1
22 ON ERROR E%,R%GOTO33
23 G=A(K)+A(I)
24 FOR P=K+1TOI-1:FOR L=P+1TOI-1:S=A(P)+A(L):IF S=GTHEN27:NEXT L
25 IF P=I-2THEN26:NEXT P
```

```
26 A(I+1)=G:GOTO 29
27 K=K+1:IF K=I-2THEN28:GOTO 23
28 A(I+1)=A(I-2)+A(I):G=A(I+1)
29 PRINT I+1,G,(I+1)/G:KEYIN Fx,44,44:I=I+1:IF I=3000THEN67:GOTO 21
30 % ВОЗВРАТ УПРАВЛЕНИЯ ИЗ ЦИКЛА:RETURN
31 DIM A(3000):DATA LOAD DC OPEN R"DATA"
32 DATA LOAD DC A():DATA SAVE DC CLOSE ALL:GOTO 15
33 PRINT "ВОЗНИКЛА ОШИБКА":DIM Cx11,Px13
34 Cx="КОД ОШИБКИ=":Px="В СТРОКЕ=":PRINT Cx;Ex;Px;Rx
35 PRINT "ВОЗМОЖНЫЕ ВАШИ ДЕЙСТВИЯ:":PRINT "**********************"
36 PRINT "ПОДГОТОВИТЬ УВВ - НАЖАТЬ (CONTINUE)"
37 PRINT "ВЫПОЛНИТЬ (RESTART),ЗАТЕМ (RUN) И (CR/LF)"
38 PRINT "ПЕРЕПИСАТЬ МАССИВ (ПООС) НА ДИСК - 1*"
39 PRINT "ПОВТОРИТЬ РАБОТУ С I-ЭЛЕМЕНТА (ПООС) -2*"
40 INPUT "ВВЕСТИ НОМЕР(*) ДЕЙСТВИЯ",L:ON LGOTO43,42
41 PRINT "ОШИБКА В ЗАКАЗЕ":PRINT HEX(0312):GOTO 35
42 INPUT "ВВЕСТИ НОМЕР ЭЛЕМЕНТА (ПООС)",Y:I=Y:GOTO 21
43 GOSUB 64:GOTO 31
44 PRINT "КОМПЬЮТЕР ОСТАНОВЛЕН И ЖДЕТ ВАШЕГО РЕШЕНИЯ:"
45 PRINT "ПЕРЕПИСАТЬ (ПООС) НА ДИСК          - 1*"
46 PRINT "ВЫЧИСЛИТЬ ДИВЕРГЕНЦИЮ ПЛОТНОСТИ    - 2*"
47 PRINT "ВЫЧИСЛИТЬ ЧИСЛО БЛИЗНЕЦОВ (ПООС)   - 3*"
48 PRINT "ПРОДОЛЖИТЬ РАБОТУ С ПРОГРАММОЙ,    - 4*"
49 PRINT "ВЫЧИСЛИТЬ ДИВЕРГЕНЦИЮ БЛИЗНЕЦОВ    - 5*"
50 PRINT "ЗАВЕРШИТЬ РАБОТУ С ПРОГРАММОЙ      - 6*"
51 INPUT "ВВЕСТИ НОМЕР(*)",W:ON WGOTO43,53,59,68,68,95
52 PRINT "ОШИБКА ЗАКАЗА":GOTO 44
53 PRINT "ДИВЕРГЕНЦИЯ ПЛОТНОСТИ В (ПООС)"
54 PRINT "НОМЕР ЭЛЕМЕНТОВ ПООС            ДИВЕРГЕНЦИЯ"
55 INPUT "ВВЕСТИ ИНТЕРВАЛ(MIN/MAX(3000) (ПООС)",Z,D
56 IF A(D)=0THEN58:FOR U=ZTOD-1:PRINT U,U+1,(U+1)/A(U+1)-U/A(U)
57 NEXT U:PRINT "ВАШ ЗАКАЗ ВЫПОЛНЕН":GOTO 44
58 PRINT "НЕДОПУСТИМЫЙ ИНТЕРВАЛ - УМЕНЬШИТЬ":GOTO 55
59 INPUT "ВВЕСТИ ИНТЕРВАЛ(MIN/MAX(3000) (ПООС)",Z,D
60 IF A(D))0THEN61:PRINT "НЕДОПУСТИМЫЙ ИНТЕРВАЛ-УМЕНЬШИТЬ":GOTO 59
61 PRINT "ЧИСЛО ПАР БЛИЗНЕЦОВ В ИНТЕРВАЛЕ":T=0:FOR U=ZTOD-1
62 IF A(U+1)-A(U))1THEN30:T=T+1:NEXT U:PRINT T:GOTO 30:GOSUB 64
63 GOTO 44
64 % ПОДПРОГРАММА ПЕРЕЗАПИСИ (ПООС) НА ДИСК:DATA LOAD DC OPEN R"DATA"
65 DATA SAVE DC #A():DATA SAVE DC CLOSE ALL:PRINT "(ПООС) ПЕРЕПИСАНА"
66 % КОНЕЦ ПРОГРАММЫ ПЕРЕЗАПИСИ НА ДИСК:RETURN
67 GOSUB 64:PRINT "ФАЙЛ ДЛЯ (ПООС) СФОРМИРОВАН НА ДИСКЕ":GOTO 35
68 INPUT "ВВЕСТИ ИНТЕРВАЛ(MIN/MAX(3000) (ПООС)",Z,D:GOTO 68
74 PRINT "ВАРИАНТ РАБОТЫ С ПООС(A,B)":DIM Ex4,Rx4
75 INPUT "ВВЕСТИ ПЕРВЫЕ 5 ЧЛЕНОВ (ПООС)",A1,A2,A3,A4,A5
76 A(1)=A1:A(2)=A2:A(3)=A3:A(4)=A4:A(5)=A5
77 PRINT "     НОМЕР   #     ЗНАЧЕНИЕ     #   ПЛОТНОСТЬ"
78 PRINT "     ЧЛЕНА   #     ЧЛЕНА        #   ОТРЕЗКА ПООС"
79 FOR I=1TO5:PRINT I,A(I),I/A(I):NEXT I:ON ERROR Ex,Rx:GOTO91
80 I=5:K=1
81 G=A(I):M=0
82 G=G+1:M=0
83 FOR P=KTOI-1
84 FOR L=P+1TOI
85 S=A(P)+A(L):IF S=GTHEN86:GOTO 87
86 M=M+1:IF M=2THEN88
87 V=1942:NEXT L:NEXT P:ON M+1GOTO82,89
88 GOTO 82
89 A(I+1)=G:PRINT I+1,A(I+1),(I+1)/A(I+1):I=I+1:K=1
90 KEYIN Fx,92,92:GOTO 81
91 PRINT "ВОЗНИКЛА ОШИБКА КОМПЬЮТЕРА"
92 PRINT "(ИСКРА 226) ОСТАНОВЛЕНА И ЖДЕТ ВАШЕГО РЕШЕНИЯ:"
93 INPUT "ПРОДОЛЖИТЬ РАБОТУ(1-ДА/2-НЕТ)",W:ON WGOTO45,95
94 PRINT "ОШИБКА ОТВЕТА":PRINT HEX(0312):GOTO 93
95 END :% КОНЕЦ ПРОГРАММЫ ДЛЯ РАБОТЫ С ВАРИАНТАМИ (ПООС)
```

Листинг программы "ПООС" ясен; он снабжен необходимыми комментариями и дополнительных пояснений не требует. Сообщения текста программы в режиме диалога содержат возможные ответы, что позволяет пользователю, знающему суть задачи, легко выбирать в той или иной момент нужный

ему ответ. Ниже приводится небольшой фрагмент результатов одного варианта вычислений по программе "ПООС":

НОМЕР ЧЛЕНА	ЗНАЧЕНИЕ ЧЛЕНА	ПЛОТНОСТЬ ОТРЕЗКА ПООС
13	36	.3611111111111
14	38	.3684210526316
15	51	.2941176470588
16	53	.3018867924528
17	61	.2786885245902
18	63	.2857142857143
19	76	.25
20	78	.2564102564103

После многочисленных расчетов по программе "ПООС" на ЭВМ "Искра-226" были получены интересные результаты, которые с успехом использованы в теоретических исследованиях ПООС (а, в) и ПООС1 (а, в). Вкратце эти результаты сводятся к следующему.

Рассмотрим сначала ПООС1 (а, в). К сожалению, алгоритм образования элементов последовательности обнаружить не удалось, так как поведение элементов последовательности довольно сложно. Опираясь на эксперименты, удалось установить, что каждая ПООС1 (а, в) имеет бесконечное множество пар близнецов, по крайней мере, одного из типов Б(а), Б(в) или Б(а + в). Доказано, что если $a_к$ есть к-й элемент ПООС1 (а, в), то к-й элемент ПООС1 (ра, рв) есть $pa_к$. Дальнейшие исследования ПООС1 (а, в) продолжаются.

Совершенно иная картина имеет место в случае ПООС (а, в). Здесь удалось получить почти исчерпывающие решения ряда вариантов задачи. Так, ПООС (1,3) имеет бесконечные множества Б(в) и Б(в + 1). Начиная с $k = 9$ остальные элементы ПООС (1,3) определяются по формулам

$$\begin{cases} a_к = 11p + 9; \\ a_{к+1} = 11p + 13; \quad p = [к/3] - 2; \\ a_{к+2} = 11p + 17; \; к = 9 + t/t = 0,1,2,\ldots . \end{cases}$$

Очевидно, что плотность ПООС (1,3) относительно множества N есть $\rho = 3/11$.

ПООС (1,4) имеет бесконечные множества Б(в) и Б(в + 1). Начиная с к=12 остальные элементы последовательности находятся по формулам

$$\begin{cases} a_к = 3p + 4к - 10; \\ a_{к+1} = 3p + 4к - 6; \; p = [к/3] - 4; \\ a_{к+2} = 3p + 4к; \; к = 12 + 3t/t = 0,1,2,\ldots . \end{cases}$$

Плотность ПООС (1,4) относительно множества N есть $\rho = 0,2$.

ПООС (1,в) при в $\geqslant 5$ имеет бесконечные множества Б(в) и

Б(в + 1). Все элементы таких ПООС вычисляются по формулам

$$a_к = \begin{cases} в + к - 2, \text{ если } к \in \{3,4,..., в + 2\}; \\ 4в - 2, \text{ если } к = в + 3; \\ (к - в + 1)в + [(к - в - 3)/2] - 2 - \text{в остальных} \end{cases}$$

случаях.

Плотность ПООС (1,в) при $в \geqslant 5$ относительно множества есть $\rho = 2/(2в + 1)$.

В ПООС (а, в) при условии, что а>1 и в/а – [в/а]>0, начиная с $к \geqslant 3$, все остальные элементы определяются по формуле

$a_к = в + (к - 2)а$.

Множество Б(а) в такой последовательности бесконечно и плотность ПООС (а,в) относительно множества N есть $\rho = 1/а$.

Подобно ПООС1 (а, в) для ПООС (а, в) справедливо следующее, если $а_к$ есть элемент ПООС (а, в), то к-й элемент ПООС (ра, рв) есть $ра_к$.

В случае ПООС (а, 2а) множества Б(2а) и Б(8а) (их плотности относительно множества N есть соответственно 0,5 и 0,25) бесконечны. Алгоритм определения элемента $а_к$ существует. В других случаях ПООС (а, в) также можно установить алгоритм формирования $а_к$ как функцию параметров к, а и в. Примеры:

ПООС (2,8). При $к \geqslant 10$ имеем:

$$\begin{cases} a_{10+к} = a_9 + 10 \cdot к + 8; \\ a_{11+к} = a_9 + 10 \cdot (к + 1); \\ a_{12+к} = a_9 + 10 \cdot (к + 1); \end{cases} \quad к = 0,3,6,9,12,... .$$

Множества Б(в), Б(а + в), Б(2а + в) при этом бесконечны;

$\rho = 1/(а + в)$!

ПООС (2,10). При $к \geqslant 10$ получаем:

$$\begin{cases} a_{10+к} = 22р + 36; \\ a_{11+к} = 22р + 46; \end{cases} \quad р = [к/2] + 1; к = 0,2,4,6,... .$$

Множества Б(а), Б(а + в) при этом бесконечны; $\rho = 2/(а + 2в) = 1/11$.

ПООС (2,6). При $к \geqslant 10$ имеем:

$$\begin{cases} a_{10+к} = 11р + 26; \\ a_{11+к} = 11р + 34; \\ a_{12+к} = 11р + 40; \end{cases} \quad р = 2\,(к/3 + 1); к = 0,3,6,9,12,... .$$

Множества Б(в), Б(а + в) при этом бесконечны;

$\rho = 3/(в + 2) = 0,375$.

ПООС (3,9). При $к \geqslant 9$ получаем:

$$\begin{cases} a_{9+к} = a_9 + 33р; \\ a_{10+к} = a_9 + 33р + 12; \end{cases} \quad р = [к/3];$$

$a_{11+к} = a_9 + 33p + 24;$ $к = 0,3,6,9,12,...$.

Множества Б(в) и Б(а + в) при этом бесконечны; $\rho = 3/(2а + 3в) = 1/11$.

ПООС (9,45). При к≥9 имеем:

$$\begin{cases} a_{9+к} = a_8 + 99p + 45; & p = [к/2]; \\ a_{10+к} = a_8 + 99p + 99; & к = 0,2,4,6,... \end{cases}$$

Множества Б(в), Б(а+в) при этом бесконечны; $\rho = 2/(2в + а) = 2/99$.

ПООС (10,30). При к≥10 получаем:

$$\begin{cases} a_{10+к} = 70p + 210; \\ a_{11+к} = 70p + 250; \\ a_{12+к} = 70p + 280; \end{cases} \quad p = к/3; \quad к = 0,3,6,9,12,... .$$

Множества Б(в) и Б (а + в) при этом бесконечны; $\rho = 3/(2а + 3в) = 3/110$.

ПООС (10,200). При к≥24 имеем:

$$\begin{cases} a_{24+к} = 410p + 980; \\ a_{25+к} = 410p + 1190; \end{cases} \quad p = [к/2]; \; к = 0,2,4,6,... .$$

Множества Б(в), Б(а + в) при этом бесконечны; $\rho = 2/(2в + в) = 1/205$.

Для описания поведения ПООС (1,2), а значит, и ПООС (а, 2а) поступаем следующим образом. Наряду с множествами К и А (номеров к и значений $a_к$ элементов последовательности) вводим множество Р разностей $a_{к+1} - a_к$ (изменение элементов ПООС). Оказывается, что структура множества Р более наглядна для исследования. Начиная с к = 14 во множестве Р прослеживается следующая интересная закономерность. Будем называть $p_к \in P$ **скачком**, если $p_к = a_{к+1} - a_к \neq 2; 8$. **Нарастающим скачком** назовем такой элемент $p_к \in P$, который максимален среди всех p_i ($i < к$). Нарастающие скачки не ограничены сверху, и их значение растет с увеличением к. Интервал из четырех элементов $\langle 2,8,2, p_к \rangle$ ($p_к$ – скачок, не обязательно нарастающий) будем называть **базовым** (В). Оказывается, что начиная с к=14 все множество Р состоит из базовых интервалов, прилегающих друг к другу, т. е.

$$P = \{\langle 2,8,2,p_1\rangle \langle 2,8,2,p_2\rangle \langle 2,8,2,p_3\rangle ... \langle 2,8,2,p_к\rangle\}.$$

Следовательно, изучение множества Р сводится к исследованию его подмножества $p_1 = \{p_к\}$ скачков. Можно показать, что распределение нарастающих скачков $p^1_к$ в Р подчиняется следующей закономерности: нарастающие скачки $p^1_к$ распределяются друг от друга через промежутки $2^{[к/2[}$ (В) членов множества Р (к=1,2,3,...).

Положив $p_0 = p_0^1 = 13$ ($13 = a_{15} - a_{14}$) первым среди множества всех нарастающих скачков, остальные можно найти по следующим реккурентным формулам:

$$\begin{cases} p_{2k-1}^1 = 3p_{2k-2}^1 - 1; \\ p_0^1 = 13;\ k = 1,2,3,\ldots(\ )^*; \\ p_{2k}^1 = 2p_{2k-1}^1 - p_{2k-2}^1; \end{cases}$$

Не нарушив общности, рассмотрим поведение элементов p_{2k}^1 ($k = 1,2,3,\ldots$). Нетрудно убедиться в том, что каждый член p_{2k}^1 отстоит от элемента p_0^1 на расстояние $L = 4(2^{k+2} - 4)$ элементов множества P или ПООС (1,2). Следовательно, p_{2k}^1 приходится на элемент $n = L + 14$ множества N.

Внутри каждого базового интервала значение элементов ПООС (1,2) возрастает на $12 + p_i$ сам же элемент p_{2k}^1, как следует из формул (*), удовлетворяет неравенству $p_{2k}^1 \leqslant 72p_\sigma^1 \times 5^{2k-10}$, а так как до нарастающего скачка p_{2k}^1 имеется L элементов множества P, то значение элементов ПООС (1,2) на этом промежутке возрастает не менее, чем на $[(12+p_0)/4] \times L + 72p_0^1 \cdot 5^{2k-10}$. Следовательно, плотность ПООС (1,2) относительно множества N

$$\lim_{k\to\infty} [4(2^{k+2}-4) + 14]/[(12 + p_0)(2^{k+2}-4) + 72p_0 \cdot 5^{2k-10}] = 0,$$

т. е. $\rho = 0$.

Более того, можно показать, что внутри интервала $\langle p_k^1, p_{k+1}^1 \rangle$ между любыми двумя нарастающими скачками скачки p_i в базовых интервалах состоят из набора скачков $\{p_k^1, p_{k+1}^1\}$ и расположены симметрично относительно центрального скачка, равного p_k^1, т. е. имеют распределение

$$\langle p_k^1 (282p_i)(282p_{i+1})\ldots(282p_k^1)\ldots(282p_i)(282p_{i+1})(282p_{k+1}^1) \rangle$$

Внутри интервала $\langle p_k^1, (282p_k^1) \rangle$ картина повторяется и т. д. Такое поведение множества P позволяет для ПООС (1,2) [а значит, и ПООС (a, 2a), где a – целое] определить значение a_k элементов последовательности как функцию параметров k и a.

Все вышеизложенное резюмирует следующее утверждение.

Утверждение 1. Каждая ПООС (a, в) имеет бесконечное множество пар близнецов, по крайней мере, одного из типов Б(a), Б(в) или Б(a + в); при этом ПООС (a, 2a) – множество Б(2a), причем плотность этой последовательности относительно множества N равна нулю. Элементы ПООС (a, в) являются функциями параметров a и k, причем для каждого целого $в \geqslant 3$ ПООС (1, в) имеет бесконечные множества Б(в), Б(в+1); элементы a_k этих последовательностей вычисляется по формулам $a_k = \Phi_3$ (k, в) при (в = 3), $a_k = \Phi_4$ (a, в) (при в = 4) и

$a_к = Ф_б$ (к, в) (при в$\geqslant$5). В ПООС (а, в) при условии, что а>1 и в/а − [в/а]>0, все элементы определяются по формуле $a_к$ = = в + (к − 2)а, где к = 3,4,5,... . Множество Б(а) в таких последовательностях бесконечно.

5.2. Задача Штейнгауза

Рассмотрим еще несколько интересных применений ЭВМ "Искра-226" для решения математических задач из теории комбинаторики и теории чисел, где без помощи ЭВМ обойтись очень трудно. Примеры программ, используемых для решения этих задач, содержат интересные фрагменты применения операторов языка БЕЙСИК.

Известным польским математиком Г. Штейнгаузом более четверти века тому назад была сформулирована до сих пор нерешенная задача "плюсы − минусы", суть которой заключается в следующем (вместо "плюс−минус" будем использовать обозначение 1/0). Имеется строка C_T из элементов 0 и 1, занимающая длину T $\in$ {3 + 4к или 4+4к/к = 0,1,2,3,... }. Элементы ее пронумеруем слева направо как р(1,1), р(1,2), ..., р(1,Т); р(1,J) $\in$ {0,1}, где J=$\overline{1,T}$. Следующую строку длиной (Т − − 1) получаем из предыдущей согласно закону

$$р(2,к) = р(1,к) + р(1,к+1) + 1 (\mathrm{mod}\ 2), \text{ где } к=\overline{1, T-1}.$$

Третья строка элементов аналогичным образом получается из второй и т. д. В итоге к−я строка элементов формируется по закону

$$р(к, i) = р(к - 1, i) + р(к - 1, i + 1) + 1 (\mathrm{mod}\ 2),$$
$$\text{где } \overline{i = 1, T - к + 1};\ к = \overline{2, T}.$$

В результате образуется треугольная фигура Δ_T, состоящая из 0 и 1. Строку C_T назовем **решением** Р(Т) задачи Штейнгауза размерности Т, если из нее генерируется треугольная фигура Δ_T, содержащая равное число Т(Т+1)/4 вхождений символов 0 и 1.

Основной теоретический результат, полученный по собственно задаче Штейнгауза, сводится к следующему. Все решения этой задачи размерностей 3-и 4 назовем базовым В, причем В = { 000,011,101,110;0011,0101,1010,1011,1100,1101 }, а решение задачи размерности Т > 4 − **производным**, если оно представимо в виде конкатенации решений $Р(Т_1)$ Р $(Т_2)$... $Р(Т_к)$ задачи размерностей $Т_i$<Т.

Базовые решения представляют особый интерес в задаче Штейнгауза, причем для любого допустимого Т$\geqslant$3 эта задача имеет непустое множество решений Р(Т), включающее в себя непустые подмножества базовых и производных решений.

Для экспериментального исследования различных аспектов поведения решений задачи Штейнгауза была разработана базовая программа "Плюсы–минусы" на языке БЕЙСИК, листинг которой имеет вид:

```
0 PRINT HEX(0312);"ЗАДАЧА (ПЛЮСЫ-МИНУСЫ)":DIM DX(80),BX(80,80)
1 PRINT "ВВЕСТИ РАЗМЕР(N) ФИГУРЫ; N ВЫБИРАЕТСЯ ТАК;"
2 PRINT "ЧТОБЫ ЧИСЛО N(N+1)/4 БЫЛО ВСЕГДА ЦЕЛЫМ; ДОСТАТОЧНО"
3 PRINT "ПОЛОЖИТЬ: N=3+4K ИЛИ N=4+4K (K=0,1,2,3,4,5,.....)"
4 INPUT "РАБОТА ПО ЗАКАЗУ(1) ИЛИ АВТОМАТИЧЕСКИ(2)",L
5 INPUT "ВВЕСТИ РАЗМЕР ЗАДАЧИ - ЧИСЛО N",N:LET W,T=0
6 PRINT HEX(030712):ON L-1GOTO21:FOR P=1TON
7 INPUT "ВВЕСТИ BX(1,N)",BX(1,P):NEXT P:GOSUB 10
8 PRINT "РАЗМЕР ЗАДАЧИ ШТЕЙНГАУЗА=";N;" ЧИСЛО РЕШЕНИЙ=";W
9 INPUT "РАБОТУ ПРОДОЛЖИТЬ(1-ДА/2-НЕТ)",L:ON LGOTO4:END
10 % ФОРМИРОВАНИЕ ТРЕУГОЛЬНЫХ ФИГУР И ИХ АНАЛИЗ:LET E1,E0=0
11 FOR I=1TON-1:FOR K=1TON-1
12 IF BX(1,K)+BX(1,K+1)=2THEN15
13 IF BX(1,K)+BX(1,K+1)=0THEN15
14 IF BX(1,K)+BX(1,K+1)=1THEN16
15 BX(I+1,K)=1:E1=E1+1:GOTO 17
16 BX(I+1,K)=0:E0=E0+1
17 NEXT K:NEXT I:FOR I=1TON:IF BX(1,I)=0THEN18:E1=E1+1:GOTO 19
18 E0=E0+1
19 NEXT I:IF ABS(E1-E0))0THEN20:W=W+1
20 PRINT HEX(0312):RETURN :% КОНЕЦ ФОРМИРОВАНИЯ И АНАЛИЗА
21 INPUT "РАБОТА ВПЕРВЫЕ(1-ДА/2-НЕТ)",L:ON LGOTO22,29
22 FOR I=1TON:DX(I)=1:NEXT I
23 FOR I=1TON:IF DX(N-I+1)=1THEN24:DX(N-I+1)=1:GOTO 25
24 DX(N-I+1)=0:NEXT I
25 KEYIN A%,26,26:GOTO 31
26 DATA LOAD DC OPEN R"RESULT":DATA SAVE DC %R,T,W,DX()
27 DATA SAVE DC CLOSE ALL:PRINT "N%  ВАРИАНТА=";T;" РЕШЕНИЙ=";W
28 INPUT "РАБОТУ ПРОДОЛЖИТЬ(1-ДА/2-НЕТ)",L:ON LGOTO31:END
29 DATA LOAD DC OPEN R"RESULT":DATA LOAD DC R,T,W,DX()
30 DATA SAVE DC CLOSE ALL
31 T=T+1:IF T)2^NTHEN8:FOR I=1TON:BX(1,I)=DX(I):NEXT I
32 GOSUB 10:GOTO 23:END :% КОНЕЦ ПРОГРАММЫ
```

Этот листинг полностью соответствует приведенной выше формулировке задачи Штейнгауза и дополнительных пояснений не требует. Многочисленные расчеты по программе "Плюсы–минусы" и ее модификациям позволили эмпирически установить ряд интересных свойств поведения решений задачи, выражаемых следующим утверждением.

Утверждение 2. Общее совокупное число $P(T)$ решений задачи Штейнгауза размерности T удовлетворяет неравенству $P(T) \gg 2^{T-f(T)}$, где $f(T) \leqslant [T/2]$.

Пусть размерность данной задачи $T \in \{к_1+4к,\ к_2+4к/к = 0,1,2,\ldots\}$. Тогда число базовых решений задачи – не менее $2^{3к-2}$ (для $к_1=3$) и $2^{3к}$ (для $к_2=4$), где $k = 1,2,3,\ldots$. Подобная картина поблюдается и для производных решений.

Число $T(к)$ решений обобщенной задачи Штейнгауза при $A = \{0,1,2\}$ и допустимых значениях $к \in \{2+3р,\ 3+3р/р=1,2,3,\ldots\}$ удовлетворяет неравенству $T(к) > 2^{к-1}$.

5.3. Задача "Взлеты и падения чисел-градин"

Еще один интересный пример использования ЭВМ "Искра-226" для решения математических задач демонстрирует занимательная задача "Взлеты и падения чисел-градин", опуб-

ликованная в американском журнале "Научная Америка" за 1984 г. Суть ее состоит в следующем.

Пусть к – целое положительное число. Каждый член P_T ($T \geqslant 0$) числовой последовательности P(к) образуется по правилу:

$$p_0 = к;\ p_{T+1} = \begin{cases} p_T/2, \text{ если } p_T \text{ – четное число;} \\ 3p_{T+1} \text{ – в противном случае,} \end{cases}$$

где T = 0,1,2,3,... . Требуется определить поведение членов (чисел-градин) последовательности P(к).

Для решения данной задачи наряду с теоретическими исследованиями [50] были использованы результаты, полученные на ЭВМ "Искра-226" с помощью простой программы: листинг которой имеет вид:

```
0 PRINT HEX(0312);" ВЗЛЕТЫ И ПАДЕНИЯ (ЧИСЕЛ-ГРАДИН)
1 PRINTUSING 0:INPUT "ВВЕСТИ ПЕРВОЕ ЧИСЛО N)0",N:M=N:T=0
2 PRINT AT(15,30);:PRINT "НОМЕР=";T;"# ЭЛЕМЕНТ=";N
3 KEYIN G#,7,7:IF ABS(N/2-INT(N/2))>0THEN5
4 T=T+1:N=N/2:GOTO 6
5 T=T+1:N=3#N+1
6 IF N()4THEN2:PRINT "M=";M;" T=";T;" N=";N:STOP
7 INPUT "РАБОТУ ПРОДОЛЖИТЬ(1-Д/2-Н)",L:ON LGOTO8:END
8 M=M+1:N=M:PRINT HEX(0312);:T=0:GOTO 3
```

Теоретически доказано [50], что начиная с произвольного целого к>0 числовая последовательность P(к) через T=T(к) шагов содержит элемент p_T, не больший 2000000. С другой стороны, результаты вычислений по приведенной программе показали, что любая числовая последовательность P(с) при $с \leqslant 2000000$ содержит элемент, равный 4. На основе этого можно сформулировать следующее утверждение.

Утверждение 3. Для каждого целого к > 0 существует целое число в = в(к) > 0 такое, что в (к)-й элемент числовой последовательности P(к) равен 4, т. е. любая числовая последовательность P(к) является периодической с периодом 3, начиная с в (к) = 20 элемента.

5.4. Моделирование параллельных динамических систем

Рассмотрим теперь возможность использования ЭВМ "Искра-226" при моделировании одного интересного класса параллельных динамических систем (ПДС) [50,51], где преимущества данной ЭВМ по сравнению с ЭВМ других типов особенно ощутимы.

Параллельные динамические системы для своего моделирования, во-первых, требуют значительной части основных ресурсов ЭВМ: ОП и процессора. Во-вторых, имея дело с параллельной обработкой многомерных слов (конфигураций) в этих системах, не обойтись без хорошо развитых программных и аппаратных средств, ориентированных на диалоговый режим работы с постоянным отображением обрабатываемой

информации (или ее части) на дисплеях различного типа. Наконец, существенные временные затраты на моделирование каждого шага таких систем делают практически нереальной организацию эффективных диалоговых симуляционных систем на основе больших ЭВМ с коллективным доступом из-за высокой стоимости машинного времени при значительных временных затратах на моделирование.

Перечисленным выше требованиям наиболее полно удовлетворяют ЭВМ класса персональных. По экспертным оценкам для моделирования ПДС наилучшим образом отвечают персональные ЭВМ с ОП не менее 512 Кбайт и быстродействием не менее 1 млн. операций в секунду. Не обладая такими техническими данными, ЭВМ "Искра-226", тем не менее, позволяет проводить ряд интересных экспериментов с симуляционными программами, моделирующими те или иные феномены ПДС, и на этой основе дает возможность эмпирически установить их характерные свойства. Ниже будут рассмотрены примеры использования этой ЭВМ при моделировании одного класса ПДС – однородных структур (ОС), которые представляют собой систему параллельной обработки информации, состоящую из взаимодействующих идентичных конечных автоматов Мура.

Однородные структуры можно интерпретировать не только как абстрацию клеточных систем, но и как формальную модель параллельной обработки информации. ОС может рассматриваться как формальная модель параллельных вычислений подобно тому, как машина Тьюринга рассматривается в качестве формальной модели вычислимости. С логической точки зрения ОС есть бесконечный автомат со специфической внутренней структурой.

Теория ОС изучает структурные и динамические свойства таких автоматов и в настоящее время составляет самостоятельную ветвь кибернетики. Особенно возрос интерес к этой теории и ее приложениям в таких областях, как параллельные алгоритмы и вычисления, ВТ параллельного действия, распознавание образов, моделирование, искусственный интеллект, математическая биология и др. [50].

Сложившаяся достаточно развитая теория, однако, не позволяет на современном этапе выявить целый ряд весьма важных динамических свойств ОС. В связи с этим возникает необходимость моделирования того или иного класса ОС на ЭВМ и эмпирического исследования поведения объекта. Для понимания последующего материала неформально определим простейший тип двухмерных ОС (2–ОС).

Пусть имеется бесконечная двухмерная регулярная целочисленная решетка P_A, в узлах которой расположено по копии конечного автомата Мура с алфавитом внутренних состояний

$A = \{0,1,...a-1\}$. Каждый автомат $x_{ij} \in P_A$ соединен одинаковым образом со своими непосредственными соседями (включая и сам автомат) $x_{i+m,j+n}(m,n \in \{0,\pm 1\})$. На таким образом заданной решетке P_A определяется локальная функция перехода Ф, которая по состояниям $S(x_{i+m,j+n}, T) \in A$ в момент времени Т автоматов, соседних с автоматом x_{ij}, определяет состояние $S(x_{ij}, T+1) \in A$ автомата x_{ij} в следующий момент времени Т+1, т. е.

$$S(x_{ij}, T+1) = \Phi[S(x_{i+m,j+n}, T)],$$

где $T \in \{0,1,2,...\}$; $n, m \in \{0 \pm$

$\pm 1\}$ и $i, j \in \{0,\pm 1,\pm 2,...\}$. При этом на локальную функцию Ф накладывается условие: $\Phi(0,T) = 0$. Любое отображение $P_A \rightarrow$ $\rightarrow A$ называется конфигурацией 2–ОС.

С практической точки зрения наибольший интерес представляют конечные конфигурации, содержащие только конечное число символов из множества $A \setminus \{0\}$. 2 – ОС функционируют в дискретные моменты времени $T>0$, переводя одну конфигурацию структуры в другую посредством одновременного применения локальной функции Ф ко всем автоматам из P_A.

Таким образом, 2–ОС является параллельным алгоритмом по переработке конфигурации (слов) в конечном алфавите А. Сама же динамика конфигурации в 2–ОС на формальном уровне описывает функционирование того или иного моделируемого в ней процесса или объекта: развитие клеточного организма, распространение импульсов в нервной ткани, параллельные вычисления, рост кристаллов и т. д. В рамках класса классических 2–ОС могут быть выделены специальные подклассы структур со специфическими свойствами, например 2–ОС с рефрактерностью (2–ОСР), определяемые следующим образом.

Алфавит 2–ОСР имеет вид $A = \{0,1\} \cup \{A,B,C,D,E,F,2,3,4,5,6,*\}$; при этом состояния 0 и 1,A,B,C, D ,E, F называются соответственно **состоянием покоя** и **состояниями возбуждения.** Автомат в состоянии покоя (0) переходит в состояние возбуждения только тогда, когда возбуждено не менее Р его соседей (порог возбудимости), а затем в одно из состояний рефрактерности глубиной М, после чего он в течение М шагов (тактов) 2 – ОСР является нефункционирующим и снова переходит в состояние покоя.

Если величина М постоянна, то имеют место 2 – ОС с фиксированной рефрактерностью; в противном случае – с переменной рефрактерностью. Естественно рассматривать глубину переменной рефрактерности в зависимости от числа импульсов, возбуждающих неактивный автомат. В состоянии * автомат называется дефектным и в функционировании 2 – ОСР участия не принимает, т. е. $\Phi(*,T) = *$. .

2 – ОСР могут служить формальной моделью ряда интересных объектов [50]. Наибольший интерес при этом представляют исследования распространения возбуждений (активности) в зависимости от типа рефрактерности, ее глубины, порога возбудимости, наличия дефектных автоматов и начальной конфигурации области возбуждения. В [27] получен ряд общих теоретических результатов для такого класса ОС, но неизученными остались более тонкие свойства 2 – ОСР. Поэтому для эмпирического исследования динамических свойств 2 – ОСР была составлена симуляционная программа на языке БЕЙСИК для ЭВМ "Искра-226", листинг которой имеет вид, приведенный на стр. 100.

Программа работает в диалоговом режиме, в ней используется ряд интересных возможностей БЕЙСИКа и она может служить хорошим примером программ такого типа. Программа позволяет моделировать 2 – ОС с переменной и фиксированной рефрактерностью. Активная область моделируемой 2 – ОСР ограничивается только объемом доступной памяти ЭВМ "Искра-226".

Пользователь в начале работы программы определяет размер активной области, режим (автоматический или диалоговый) вывода результатов моделирования 2 – ОСР, тип (фиксированная или переменная) рефрактерности. Затем он задает размер окна, в которое заносит начальную конфигурацию структуры, саму конфигурацию, формируемую в режиме EDIT, и помещает конфигурацию окна в активную область моделируемой 2 – ОСР согласно задаваемым координатам.

Пользователь может неоднократно формировать окна и заносить их содержимое в нужные места активной области 2 – ОСР. После этого он задает порог возбудимости и (или) глубину рефрактерности и выводит начальную конфигурацию активной области полностью (на принтер или дисплей) либо постранично (размером 20×80 символов) на дисплей.

В автоматическом режиме работы ЭВМ после моделирования очередного шага 2 – ОСР программа выводит конфигурацию активной области на печать (принтер или дисплей по желанию пользователя) и можно переходить к моделированию следующего шага структуры. При этом пользователь нажатием любой клавиши клавиатуры может вывести ЭВМ из автоматического режима.

В режиме диалога с ЭВМ после вывода результатов моделирования очередного шага 2 – ОСР пользователь имеет возможность: продолжить моделирование следующего шага структуры; вывести результаты на печать (принтер или дисплей); изменить состояние режима моделирования (конфигурацию

```
1 COM M%(162,162)1,C%(8),L%80,P%160,Z%(2,13)1,N%1,O%(162,162)1,S%(20,20)1,S9%20
2 PRINT HEX(0312);TAB(18);"SIMULATION PROGRAM FOR REFRACTORITY STRUCTURES"
3 PRINT :PRINT :INIT (00)M%(),C%(),L%,P%,O%():G%='(FEFRACT. FIXED)'
4 INPUT "DEFINE ACTIVE 2-HSR ARRAY M%(A,A) I3(A(163)",A:P,M,G=0
5 INPUT "OUTPUT OF GENERATION RESULTS IN AUTOMATIC(1) OR DIALOG(2) MANNER",U
6 INPUT "FIXED(1) OR VARIABLE(2) REFRACTORITY BEHAVIOUR",D1:INIT (00)S%()
7 GOSUB ' 1(D1):GOSUB ' 3:GOSUB 39:GOSUB 52:ON D1-1GOTO24
8 REM GENERATION FIXED REFRACTORITY 2-HSR:KEYIN F%,67,67
9 INPUT %T1:FOR I=2TOA-1:FOR J=2TOA-1:IF M%(I,J)='*'THEN13:IF M%(I,J))'1'THEN14
10 IF M%(I,J)='1'THEN16:T=0
11 FOR K=I-1TOI+1:FOR L=J-1TOJ+1:IF M%(K,L)()'1'THEN12:T=T+1
12 NEXT L:NEXT K:IF T(PTHEN17:O%(I,J)='1':GOTO 17
13 O%(I,J)=M%(I,J):GOTO 17:DEFFN ' 4:INPUT "INPUT SIZE H(21",H:MAT REDIM S%(H,H)1:RETURN
14 CONVERT M%(I,J)TOZ:Z=Z-1:IF Z)2THEN15:Z=0
15 CONVERT ZTON%,(0):O%(I,J)=N%:GOTO 17
16 CONVERT NTON%,(0):O%(I,J)=N%:GOTO 17
17 NEXT J:NEXT I:G=G+1:GOSUB ' 0:INPUT %T2:GOSUB  2:INIT (00)O%():PRINT "STEP ";G;'T=';V1
18 IF U=2THEN19:GOSUB 55:IF G)=INT((A-2)/2)THEN67:GOTO 8
19 INPUT "GENERATION(1),VIEWING(2),ALTERING(3),END(4)",D
20 ON DGOTO8,21,22:GOTO 67
21 GOSUB 52:GOTO 19:DEFFN ' 2:V1=ROUND((T1-T2)/120000,1):RETURN
22 GOSUB 39:GOTO 19:DEFFN ' 1(B):IF B=1THEN23:STR(O%,11,5)='VARBL
23 RETURN :REM БЛОК ГЕНЕРАЦИИ 2-HSR С ПЕРЕМЕННОЙ РЕФРАКТЕРНОСТЬЮ
24 KEYIN F%,67,67:W1%="ABCDEF123456*":W2%="123456012345*":FOR K=1TO13
25 Z%(1,K)=STR(W1%,K,1):Z%(2,K)=STR(W2%,K,1):NEXT K:G=0
26 INPUT %T1:FOR I=2TOA-1:FOR J=2TOA-1:FOR L=1TO13:IF M%(I,J)=Z%(1,L)THEN27:NEXT L:GOTO 28
27 O%(I,J)=Z%(2,L):GOTO 33
28 T=0:FOR K=I-1TOI+1:FOR L=J-1TOJ+1:FOR V=1TO6
29 IF M%(K,L)()Z%(1,V)THEN30:T=T+1:GOTO 31
30 NEXT V:GOTO 31
31 NEXT L:NEXT K:IF T(PTHEN33
32 O%(I,J)=Z%(1,T)
33 NEXT J:NEXT I:G=G+1:GOSUB  0:INPUT %T2:GOSUB ' 2:INIT (00)O%():PRINT "STEP ";G;'T=';V1
34 KEYIN F%,67,67:IF U=2THEN35:GOSUB 55:IF G)=INT((A-2)/2)THEN67:GOTO 26
35 INPUT "GENERATION(1),VIEWING(2),ALTERING(3),END(4)",D
36 ON DGOTO26,37,38:GOTO 67
37 GOSUB 52:GOTO 35
38 GOSUB 39:GOTO 35:% БЛОК ИЗМЕНЕНИЯ ПОЛНОГО СОСТОЯНИЯ 2-HSR
39 INPUT "INPUT THRESHOLD OR/AND REFRACTORITY DEEP(1),ALTERING(2),END(3)",D
40 ON DGOTO45,41:GOTO 46
41 INPUT "DEFINE COORDINATES AND STATE AUTOMATA",I,J,N%
42 IF I(1ORI)160ORJ(1ORJ)160THEN41
43 IF N%="0"ORN%="1"ORN%="*"ORN%="A"ORN%="B"ORN%="C"ORN%="D"ORN%="E"ORN%="F"THEN44:GOTO 41
44 M%(I,J)=N%:GOTO 39
45 INPUT "DEFINE DIFFERENCE FOR P OR/AND M",P1,M1:P=P+P1:M=M+M1:GOTO 39
46 RETURN :DEFFN ' 3:PRINT "FORM WINDOW OF SIZE S%(H,H),END -'?'";:GOSUB ' 4
47 FOR K=1TOH:INIT (',')S9%:PRINT "WINDOW'S STRING=";K;:LINPUT STR(S9%,1,H)
48 IF STR(S9%,1,H)='?'THEN49:STR(S%(),(K-1)*H+1,H)=STR(S9%,1,H):MAT PRINT S%;:NEXT K
49 MAT PRINT S%;:INPUT "DEFINE ORIGIN IN ACTIVE ARRAY FOR WINDOW",K,L
50 FOR I=KTOK+H-1:FOR J=LTOL+H-1:M%(I,J)=S%(I-K+1,J-L+1):NEXT J:NEXT I:INIT (' ')S%
51 INPUT "CONTINUE FORMATION OF WINDOWS(1-Y/2-N)",D:ON DGOTO47:RETURN
52 % @@KF-BWBO@@0:INPUT "CONFIGURATION OUTPUT ON DISPLAY(1) OR PRINTER(2)",D
53 ON DGOTO59,54:GOTO 65
54 STOP "CONNECT PRINTER":SELECT PRINT@C(160)
55 PRINT "G"y'ENERATION ';"KF";'-STEP ';G;G%;"A=";A;'*THRESHOLD=';P;'& DEEP=';M
56 PRINT :FOR I=2TOA-1:FOR J=2TOA-1:STR(P%,J-1,1)=M%(I,J):NEXT J
57 FOR K=1TOA-2:IF STR(P%,K,1)()'0'THEN58:STR(P%,K,1)=HEX(20)
58 NEXT K:PRINT STR(P%,1,A-2):NEXT I:IF U=1THEN66:SELECT PRINT05(80):GOTO 66
59 PRINT HEX(0312):I1,J1=2:I2=21:J2=81:T@=0:W=1
60 T@=T@+1:PRINT "AUTOMATA STATES IN THE SECTOR  N% ";T@
61 FOR I=I1TOI2:FOR J=J1TOJ2:STR(L%,J-W,1)=M%(I,J)
62 NEXT J:PRINT L%:NEXT I:INPUT "CONTINUE(1),END(2)",D:ON D-1GOTO66
63 I1=I1+20:I2=I2+20:IF I2)161THEN64:GOTO 60
64 IF T@=16THEN65:J1=82:J2=161:I1=2:I2=21:W=81:GOTO 60
65 INPUT "OUTPUT 2-HSR CONFIGURATION ON PRINTER(1-Y/2-N)",D:ON DGOTO55
66 RETURN :DEFFN ' 0:FOR K=1TOA:FOR L=1TOA:M%(K,L)=O%(K,L):NEXT L:NEXT K:RETURN
67 SELECT PRINT05(80):INPUT "CONTINUE EXPERIMENTS WITH (2-HSR) PROGRAM(1-Y/2-N)",D
68 ON DGOTO3:PRINT TAB(20);"END OF EXPERIMENTS WITH (2-HSR) PROGRAM":END
```

активной области, порог возбудимости, глубину рефрактерности) или завершить моделирование заданной 2 – ОСР. Время моделирования каждого шага 2 – ОСР любого типа хронометрируется и выводится на печать в минутах. На печать выводится и другая полезная информация о моделируемой 2 – ОСР.

После завершения моделирования данного типа 2 – ОСР пользователь может либо продолжить работу по моделированию другого типа 2 – ОСР, либо прекратить ее. Программа позволяет проводить широкий круг экспериментов по эмпирическому исследованию 2 – ОС указанных типов. Ее текст компактен, снабжен необходимыми комментариями и совместно с приведенным описанием дает возможность читателю использовать программу в том или ином качестве. Многочисленное ее использование показало эффективность программы и позволило установить ряд интересных эмпирических свойств ОСР.

Основные эмпирические результаты сводятся к следующему (P – порог возбудимости, M – глубина рефрактерности):

1. В 2 – ОСР существуют самовоспроизводящиеся в смысле Мура области активных автоматов достаточно сложной конфигурации.

2. Глубина рефрактерности сильно влияет на распространение активности в 2 – ОСР. Например, ряд самовоспроизводящихся при P=2, M=2 и 3 областей активности при M=4 становятся исчезающими (затухающими).

3. В 2 – ОСР из целого ряда начальных конфигураций при P=2, M=2 и 3 генерируются самоусложняющиеся области активности, которые при M=4 становятся затухающими.

4. Фронт возбуждения активности в 2 – ОСР задерживается "решетом" из дефектных автоматов, размер ячеек которого меньше P, и продолжает распространяться при размере ячеек, не меньшем чем P.

5. В 2 – ОСР области активности, достаточно плотно упакованные возбужденными автоматами, могут на своих границах расширяться при затухании активности внутри области.

6. В классе периодических областей активности существуют начальные области активности, которые, начиная с некоторого шага K становятся периодичными с периодом C=3 и $K \geqslant H$, где H – размер стороны минимального квадрата, содержащего начальную область активности.

7. При переменной рефрактерности картина распространения активности в 2 – ОСР существенно усложняется и требует для исследования высокопроизводительных ЭВМ.

Эмпирическое исследование 2 – ОСР с использованием ЭВМ "Искра-226" продолжается и можно надеяться на получение новых интересных результатов.

С другой стороны, моделирование именно такого типа дискретных динамических систем в определенной мере определяет границы применимости этой ЭВМ для решения задач моделирования. Числовую оценку времени моделирования одного шага 2 – ОСР (да и вообще любой 2 – ОС) позво-

ляет получить следующая простая программа, составленная на языке БЕЙСИК:

```
1 % ПОЛУЧЕНИЕ НИЖНЕЙ ОЦЕНКИ ДЛЯ ВРЕМЕНИ ВЫПОЛНЕНИЯ ШАГА 2-HS
2 COM A%(172,172)1,B%(172,172)1,R(2,40),N%1,M,T:GOTO 9
3 INPUT %T1:FOR I=2TOM-1:FOR J=2TOM-1:FOR K=I-1TOI+1:FOR L=J-1TOJ+1
4 GOSUB ' 0:NEXT L:NEXT K:NEXT J:NEXT I:FOR I=1TOM:FOR J=1TOM
5 A%(I,J)=B%(I,J):NEXT J:NEXT I:INPUT %T2:GOSUB ' 1:PRINT "M=";M
6 KEYIN N%,9,9:GOTO 3:DEFFN ' 0:% ИМИТАЦИЯ ЛОКАЛЬНОЙ ФУНКЦИИ:RETURN
7 PRINT /0C,"ЗАВИСИМОСТЬ ВРЕМЕНИ РАСЧЕТА ШАГА 2-HSR ОТ ВЕЛИЧИНЫ M"
8 PRINT /0C:FOR K=1TOT:PRINT /0G,'T=',R(2,K),"M=",R(1,K):NEXT K:END
9 INPUT "РАСЧЕТ ПРОДОЛЖИТЬ(1-Д/2-Н)",L:ON LGOTO10,7
10 INPUT "ЗАДАТЬ РАЗМЕР АКТИВНОЙ ОБЛАСТИ(M(173) И ШАГ ПРИРАЩЕНИЯ",M,D
11 T=0:GOTO 3:DEFFN ' 1:T=T+1:M=M+D:IF M>172THEN7:R(1,T)=M
12 R(2,T)=ROUND((T1-T2)/120000,1):RETURN :% КОНЕЦ ПРОГРАММЫ
```

После выполнения программы получаем зависимость времени имитации одного шага некоторой абстрактной 2 – ОС от размера активной области структуры. В процессе имитации можно усложнять локальную функцию перехода 2 – ОС, что дает возможность оценивать влияние этого фактора на время имитации шага структуры.

На основе полученных данных можно эмпирически оценить время T выполнения одного шага 2 – ОС в зависимости от размера M активной области в виде $T(M) \sim @M^2$, где @ – числовая функция, растущая с увеличением сложности локальной функции Ф.

Таким образом, задача моделирования K-мерных ОС на ЭВМ относится к классу задач так называемой NP-сложности, поскольку время моделирования K–ОС определяется соотношением $T(M) \sim @\ M^k (K \geqslant 2)$.

Следовательно, применение ЭВМ "Искра-226" для моделирования 2 – ОС приводит к парадоксу: с одной стороны, инструментальные средства языка БЕЙСИК хорошо отвечают проблеме создания диалоговых программных систем, моделирующих дискретные динамические системы, подобные 2 – ОС, а с другой – необходимость использования большего размера активных областей моделируемых 2 – ОС для исследования в них наиболее интересных поведенческих свойств приводит к весьма большим временным издержкам моделирования каждого шага 2 – ОС (в случае прослеживания поведения 2 – ОС на сотни, а то и тысячи шагов). При этом теряется сама суть диалогового моделирования.

Поэтому ЭВМ "Искра-226" можно ограниченно использовать для моделирования дискретных ПДС подобного типа. И дело здесь не столько в ограниченных ресурсах (производительности и объеме памяти) ЭВМ, сколько в том, что для моделирования подобных систем надо переходить от вычислительных средств последовательного действия к параллельным вычислительным средствам на основе однородных вычислительных структур, формальной вычислительной моделью которых и являются 2 – ОС [45].

Приведенные ниже примеры оформления программ на языке БЕЙСИК наряду с другими примерами, приведенными выше, позволяют не только продемонстрировать использование тех или иных средств в вычислительных целях, но и в определенной мере носят методический и методологический характер, обучая навыкам работы в операционной среде БЕЙСИКа. Кроме того, некоторые из приведенных примеров носят самостоятельный характер и могут быть использованы непосредственно в практической деятельности как готовые сервисные или системные средства. Читателю рекомендуется подробно разобраться со всеми этими примерами программирования в среде языка БЕЙСИК.

6. Некоторые особенности работы с ЭВМ "Искра-226"

Рассмотрим некоторые особенности работы с ЭВМ "Искра-226", с которыми пришлось столкнуться при ее использовании. Прежде всего из-за отсутствия стандартно поставляемых сервисных средств программиста по мере необходимости понадобилось разрабатывать свои собственные, а из-за отсутствия подробного описания СПО – выяснять вопросы его организации методом декодирования и проб. Остановимся в первую очередь на структуре файлов, заносимых на диски, поскольку значительная часть подготовительной работы приходится именно на работу с файловым хозяйством.

6.1. Каталогизация файлов

Каталогизированные файлы (программы ПФ или данных ФД) имеют на дисках разную структуру.

Для ПФ: первый сектор содержит в первых 10 байтах ⟨01 (признак ПФ)⟩ ⟨имя программы⟩ ⟨ПЗ⟩, где ПЗ – признак защиты = { 20 (␣), 21 (!) – без защиты; 24 ($), 25 (%) – с защитой }. Остальное содержимое первого сектора составляют HEX-нули. Сам текст программы начинается со второго сектора ПФ. Последний признак конца файла (IC), во втором и третьем байтах – число реально занятых секторов под программу, остальные байты имеют HEX-нули.

Для ФД: первый байт содержит признак данных (02), а затем следуют непосредственно данные. Структура последнего сектора ФД совпадает со структурой последнего сектора ПФ.

Каждая метка файла (кроме метки каталога) в УК имеет следующую структуру: ⟨ПрФ (2 байта)⟩ ⟨начальный сектор (2 байта)⟩ ⟨конечный сектор (2 байта)⟩ ⟨HEX-нули (2 байта)⟩ ⟨имя файла (8 байтов)⟩, где ПрФ – признак файла {1080 – ПФ; 1180 – ликвидируемый ПФ; 1100 – ликвидируемый ФД; 1000 – ФД }. Следующие 4 байта отводятся под адреса начального и

конечного секторов файла, последние 8 байтов – под его имя. Метка каталога является самой первой меткой на нулевом секторе и содержит тоже 16 байтов: первые ее 6 байтов отведены под описание каталога (2 байта – размер УК; 2 байта – адрес текущего конца каталога; 2 байта – адрес конца каталога в томе), остальные 10 байтов заполнены HEX-нулями.

При создании многотомных файлов, располагающихся на р⩾⩾2 дисках любого типа, возникает проблема идентификации накопителей, содержащих эти файлы. Стандартно поставляемое СПО таких возможностей не предоставляет. На примере использования НГМД рассмотрим один простой подход к реализации системы отслеживания многотомных файлов для ЭВМ "Искра-226" в операционной среде БЕЙСИКа.

Нулевой или последний сектор НГМД (последний, 1000-й, сектор выбирается, если файлы диска будут использоваться как в режиме каталога, так и в режиме адресации секторов) отводится под оглавление тома, состоящего из записи двух типов: записи тома и записей файлов. Запись тома является первой на секторе оглавления тома и содержит имя диска (шесть байтов). Далее на этом же секторе одна за другой следуют записи файлов 1–10, длина каждого из которых составляет 25 байтов. Их смысловое содержание раскрыто в табл. 8.

Таблица 8. Реквизиты, используемые для записи файлов

Реквизит	Длина в байтах	Смысловое содержание
1	8	Имя файла
2	4	Начальный адрес файла на диске
3	4	Конечный адрес
4	6	Имя следующего диска
5	1	Тип файла
6	2	Резервные байты

Первые три реквизита записи файла ясны и пояснений не требуют. Реквизит 5 определяет тип файла и содержит один из следующих признаков: А – файл полностью располагается в указанных пределах (однотомный файл); НСК – начальный, промежуточный и последний сегменты многотомного файла соответственно. В случае признаков Н и С реквизит 4 содержит имя диска, в котором помещен следующий за данным сегмент файла. Последние 2 байта записи отводятся под резерв и служат для дальнейшего расширения операций обмена с многотомными файлами.

6.2. Разработка программного обеспечения

Большинство программ для ЭВМ "Искра-226" оформляются в виде единого модуля, полностью располагающегося в памяти ЭВМ. Однако сравнительно небольшой объем ее ОП (64 Кбайт), предоставляемый пользователю, требует достаточно сложной модульной организации программ с поочередным использованием ОП.

В многомодульных программных средствах можно выделить следующие три основные структуры управления модулями: 1) цепную; 2) динамическую; 3) смешанную. Рассмотрим в отдельности каждую из них.

Цепная структура управления. Этой структурой реализуется следующий алгоритм управления: каждый очередной модуль задачи, находящийся в памяти ЭВМ, после выполнения заменяется следующим модулем с передачей ему необходимой информации и управления и так до конца решения всей многомодульной задачи.

Такого типа структуры управления имеют место, когда в алгоритме задачи пользователя можно выделить самостоятельные функциональные части с четко определенной последовательностью их выполнения.

В качестве примера такой структуры рассмотрим простой фрагмент программы, когда в режиме прямой адресации секторов сформированное содержимое символьной переменной A¤ записывается в заданный сектор указанного дискового тома; затем содержимое этого же сектора диска считывается в очищенную переменную A¤ и выводится на печать. Параллельно на дисплей выводятся заданные адреса сектора и дискового тома.

Реализация фрагмента программы состоит из трех модулей задачи (16.11/01,16.11/02 и 16.11/03), выполняющихся последовательно один за другим в порядке их перечисления. В первом модуле основные переменные, используемые во всех трех модулях задачи, объявляются общими и в заданный сектор с адресом 1000 выбранного тома на диск 18R записывается содержимое символьной переменной A¤, т. е. INIT (HEX(41))A¤. На печать выводится листинг данного модуля и определяется имя следующего, выполняемого за ним, модуля 16.11/02, который и загружается после выполнения первого модуля.

По второму модулю записанная (в сектор с адресом 1000 диска 18R) информация считывается в предварительно очищенную от предыдущей информации переменную A¤, выводится на печать листинг модуля, определяется имя следующего, выполняемого за ним, модуля 16.11/03, который загружается с передачей ему значений переменных и управления. По

последнему модулю выводится на печать листинг и печатается содержимое символьной переменной A¤, сформированной предыдущим модулем.

Задача в целом хронометрируется, о чем выдается информация (в минутах) на печать:

```
10 REM ЦЕПНАЯ СТРУКТУРА:COM A¤X253,D¤X3,S,B¤X19,C¤,T:INPUT ¤T
25 C¤="120 LOAD DCF P¤":INPUT "ЗАДАТЬ ВХОДНОЙ ДИСК И СЕКТОР",D¤,S
35 STR(C¤,16,1)=HEX(85)
40 B¤="60 SELECT DISK 18F":STR(B¤,19,1)=HEX(85)
50 STR(B¤,16,3)=D¤:LIST /0C:LOAD B¤60,60,60
70 INIT (HEX(41))A¤:IF STR(D¤,3,1)="F"THEN90
80 DATA SAVE DA R¤(S)A¤:GOTO 100
90 DATA SAVE BA F¤(S)A¤
100 PRINT /0C,STR(A¤,1,50):PRINT S,D¤:P¤="16.11/02"
110 STR(C¤,12,1)=STR(D¤,3,1):LOAD C¤120,120,120
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
10 STR(B¤,16,3)=D¤:STR(B¤,1,2)="20":INIT (HEX(20))A¤
30 STR(C¤,1,3)="080":STR(C¤,12,1)=STR(D¤,3,1)
40 P¤="16.11/03":IF STR(D¤,3,1)="F"THEN60
50 DATA LOAD BA R(S)A¤:GOTO 70
60 DATA LOAD BA F(S)A¤
70 LIST /0C:PRINT S,D¤:LOAD C¤80,80,80
СОДЕРЖИМОЕ СЕКТОРА  900  ДИСКА 18F
--------------------------------

БАЙТЫ 001-050: AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
БАЙТЫ 051-100: AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
БАЙТЫ 101-150: AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
ВРЕМЯ= .3
КОНЕЦ РАБОТЫ ПОСЛЕДНЕГО МОДУЛЯ
5 PRINT HEX(030712):SELECT PRINT0C(164):LIST
10 PRINT "СОДЕРЖИМОЕ СЕКТОРА ";S;" ДИСКА ";D¤
20 PRINT "--------------------------------"
30 PRINT "БАЙТЫ 001-050: ";STR(A¤,1,50)
40 PRINT "БАЙТЫ 051-100: ";STR(A¤,51,50)
50 PRINT "БАЙТЫ 101-150: ";STR(A¤,101,50)
60 INPUT ¤T1:PRINT "ВРЕМЯ=";ROUND((T-T1)/120000,2)
70 PRINT "КОНЕЦ РАБОТЫ ПОСЛЕДНЕГО МОДУЛЯ":LIST /0C
```

Динамическая структура управления. В данной структуре порядок загрузки модулей задачи на выполнение в память ЭВМ определяется некоторым условием, удовлетворяемым в процессе функционирования текущего модуля (запрос пользователя, получение определенного результата, истечение временного промежутка и т. д.). В этом случае, как правило, целесообразна разработка диспетчера, постоянно находящегося в памяти и отслеживающего весь ход решения задачи.

Типичным образом организации программ динамической структуры управления является ПО задачи "Метролог" [46]. Рассмотрим один простой пример, иллюстрирующий сказанное.

Диспетчер программы с именем 21.11/00 на основе вводимой пользователем информации генерирует свой текст на выполнение требуемой функции обслуживания (посредством оператора LOAD), затем выводит свой листинг на печать и считывает нулевой сектор заданного диска в символьный массив A¤( ). В нулевом секторе диска отыскивается первая метка программы и на основе ее содержимого определяются имя программы и адрес ее начального сектора в томе.

После этого с помощью оператора LOAD DA программа загружается в память ЭВМ (не затрагивая строк диспетчера) и управление снова получает диспетчер (строка 9140), который

печатает имя загруженной программы и ее первую ПС. Затем ищется следующая программа на нулевом секторе УК тома, описанная выше процедура повторяется и так до исчерпания нулевого сектора.

Окончив работу, диспетчер выдает об этом сообщение на печать с подсчетом затраченного времени. Сам диспетчер занимает строки, начиная с номера 9000, а все загружаемые и обрабатываемые им модули задачи занимают ПС с номерами 0–8999.

На практике описанная организация управления при решении многомодульных задач зарекомендовала себя с наилучшей стороны. Действительно, кроме обслуживания процесса выполнения задачи, диспетчер может содержать модули, реализующие наиболее часто используемые процедуры (обмен информацией с дисками, обработка особых ситуаций, вычисление функций или таблиц, ввод и контроль данных и т. п.), которые встречаются во всех или в большинстве динамически загружаемых модулях задачи.

Такой подход в большинстве случаев позволяет не только существенно ослабить требования к объему памяти ЭВМ, но и уменьшить временные затраты на решение задачи, а также значительно повысить уровень автоматизации проектирования и разработки больших многомодульных программных средств для ЭВМ "Искра-226". Наглядным тому примером служит создание ПО для решения задачи "Метролог" [46].

Листинг работы диспетчера в рассмотренном примере представлен ниже.

Текст листинга ясен и особых пояснений не требует.

Смешанная структура управления. Эта структура является комбинацией рассмотренных выше цепной и динамических структур управления процессом решения многомодульных задач. Приведенные примеры позволяют читателю легко представить себе реализацию смешанной структуры и рассмотреть несложные примеры.

```
9000 COM T1,T2,A%(256)1,D%3:INPUT %T1
9010 INPUT "ЗАДАТЬ ДИСК С ПРОГРАММАМИ",D%:DIM B0%21,C0%29
9020 B0%="9040 SELECT DISK 1CF":STR(B0%,21,1)=HEX(85):STR(B0%,18,3)=D%
9030 T=0:U=17:C0%="9140 LOAD DA T(A)0,8990,9150":LOAD B0%9040,9040,9040
9040 SELECT DISK1CR
9050 STR(C0%,29,1)=HEX(85):LOAD C0%9140,9140,9060
9060 LIST /0C:DATA LOAD BA T(0)A%()
9065 FOR K=UTO239STEP16
9070 IF STR(A%(),K,2)=HEX(1080)THEN9130:NEXT K:PRINT "МЕТОК НЕТ":GOTO 9210
9130 GOSUB ' 42(K+2)
9140 LOAD DA T(A)0,8990,9150
9150 PRINT /0C,"ИМЯ ЗАГРУЖЕННОЙ ПРОГРАММЫ: ";STR(A%(),K+8,8):LIST /0C,10
9160 U=K+16:GOTO 9065:DEFFN ' 42(N):A%=STR(A%(),N,1):GOSUB 9200
9170 B1=L:B2=P1:A%=STR(A%(),N+1,1):GOSUB 9200
9180 B3=L:B4=P1:A=B1*16^3+B2*16^2+B3*16+B4:RETURN
9200 M=VAL(A%):L=INT(M/16):P=M/16-L:P1=ROUND(P*16,0):RETURN
9210 INPUT %T2:PRINT /0C,"ВРЕМЯ=";RCUND((T1-T2)/120000,1);" МИН"

ИМЯ ЗАГРУЖЕННОЙ ПРОГРАММЫ:
10 REM ПРОГРАММА (STATUS) # ALADYEV # СКБ МПСМ ЗССР:PRINT HEX(0312)
ВРЕМЯ= .2 МИН
```

При разработке и организации ПО конкретных прикладных задач пользователь должен предусмотреть решение ряда вопросов, одним из которых является вопрос идентификации магнитных накопителей, содержащих информационную базу задачи. Этот вопрос наиболее актуален, когда информационная база, необходимая для решения задачи, находится в нескольких томах, а доступ к ней требует и обменных операций в режиме прямой адресации секторов. Идентификация томов позволяет не только идентифицировать текущие тома, но и сообщать пользователю о необходимости реконфигурации томов в соответствии с алгоритмом решения задачи, что в определенной степени обеспечивает защиту тома от несанкционированного доступа. Рассмотрим простой пример организации идентификации томов прямого доступа, использованной в задачах [30, 46]

В технической документации для ЭВМ "Искра-226" не рекомендуется использовать последние секторы дисков для хранения информации, так как на них не гарантируются высокие достоверность и качество хранения информации. Однако опыт работы с дисками показал, что в режиме прямой адресации секторов эти секторы вполне пригодны для хранения небольших объемов информации, которую можно дублировать в нескольких местах последнего сектора диска (что резко повышает надежность ее хранения). Поэтому для записи информации, идентифицирующей том, был выбран именно этот сектор диска любого типа, что позволило использовать остальные его секторы в режиме каталога файлов и (или) прямой адресации.

Имя диска (8 байтов) помещается в байты 1–8 и 128–135 сектора 1000 (для ФАУ-18) и сектора 9791 (для ФАУ-1С). Для идентификации конкретного тома по его имени предлагается простая процедура DEFFN' 86(D¤,I¤) с явными и неявными параметрами:

D¤ – адрес дискового тома (18F/18R/1CF/1CR);

I¤ – имя дискового тома (восемь символов);

R¤ – флажок идентификации (полагается равным 2, если заданное имя совпадает с именем тома, и 1 – в противном случае; параметр является неявным).

Пример использования данной процедуры приводится на с. 109.

Кроме имени тома, на последний сектор диска может быть записана и другая служебная информация пользователя, которую впоследствии можно обрабатывать. Текст приводимого листинга ясен и особого пояснения не требует. На запрос в цикле пользователь отвечает адресом диска и его именем в зависимости от того, совпадают или нет введенное имя и имя тома.

```
0 % ПРОЦЕДУРА ИДЕНТИФИКАЦИИ ДИСКОВОГО ТОМА
1 INPUT "ВВЕСТИ ДИСК И ИМЯ ТОМА",T%,F%
2 GOSUB ' 86(T%,F%):ON R0GOTO4,3:END
3 PRINT /0C,"ИМЯ ТОМА:    ";F%:GOTO 1
4 PRINT /0C,"ИМЯ ТОМА НЕ: ";F%:GOTO 1
10 DIM D%3,I%8:DEFFN ' 86(D%,I%):DIM A%(256)1
20 IF STR(D%,1,2)="18"THEN30:SELECT #11C:S0=9791:GOSUB 80:GOTO 40
30 SELECT 0118:S0=1000:GOSUB 80
40 IF STR(A%(),1,8)=I%ORSTR(A%(),128,8)=I%THEN60
50 R0=1:GOTO 70
60 R0=2:GOTO 70
70 RETURN
80 IF STR(D%,3,1)="F"THEN90:DATA LOAD BA R01,(S0)A%():GOTO 100
90 DATA LOAD BA F01,(S0)A%()
100 RETURN

ИМЯ ТОМА НЕ: INFOBASE
```

В ряде случаев использование режима прямой адресации секторов дает возможность наиболее эффективно организовывать информационную базу для задач пользователя, хотя при этом ему и не предоставляется тот сервис, которым он располагает в режиме каталога файлов, и всю работу по ведению файлового хозяйства пользователь должен полностью взять на себя.

Одним из примеров использования информационной базы в режиме прямой адресации секторов может служить организация ее в задаче "Метролог" [46], позволившая создать очень гибкую систему управления данными с быстрым к ним доступом.

Здесь на простом примере продемонстрируем принцип ведения каталога файлов в томе, который позволяет автоматизировать процесс создания каталога и доступа к файлам на диске в режиме прямой адресации секторов. Чтобы не затенять саму суть организации каталога, не будем усложнять пример общностью его применения (диск, объект обмена и режим адресации секторов будут вполне конкретными). В качестве диска выбирается резидентный диск 18F, метод доступа к дискам – режим прямой адресации типа DA.

Для реализации обмена информацией с диском через каталог файлов последний вводится в виде одномерного символьного массива B¤ (20) 42, первоначальное содержание которого имеет вид:

```
КАТАЛОГ ФАЙЛОВ НА ТОМЕ (ИНФОБАЗА)
: S&C : 0000 : 0004 : ИМЯ И КАТАЛОГ ТОМА :
: FFF : 0005 : 1600 :    EMPTY SPACE     :
```

Каждая строка каталога содержит информацию о файле, указанную в табл. 9, причем первая строка описывает нулевой сектор тома с именем тома и сам каталог в томе. Первая свободная строка каталога в байтах 3–5 содержит идентификатор FFF. Для обмена информацией с томом через каталог имеются две процедуры с формальными параметрами DEFFN'42 и DEFFN'47, позволяющие соответственно записывать (считывать) одномерные числовые массивы M ( ) размерности N под

именем F¤на (с) диск (а) с именем D¤ через каталог тома B¤ ( ). Признак PO (неявный параметр) характеризует результат завершения процедуры обмена (табл. 10).

Таблица 9. Первоначальное содержание символьного массива B¤(20)42

Байты строки	Смысловое содержание
3–5	Имя файла в томе (3 байта)
9–12	Начальный номер сектора файла (4 байта)
16–19	Конечный номер сектора файла (4 байта)
22–41	Краткая характеристика файла (18 байтов)

Таблица 10. Характеристика признака PO

Значение PO	Результат завершения процедуры обмена
0	Процедура завершена успешно
1	Записанный файл заполнил каталог тома
2	Имя тома отличается от заданного пользователем
3	Каталог для записи нового файла
4	Файл с данным именем в каталоге отсутствует

При записи файла в том через каталог пользователь имеет возможность через параметр T¤процедуры записи сформировать в каталоге краткое описание создаваемого файла (в рассматриваемом случае одномерно числового массива – длиной не более 1000 элементов).

Следовательно, имеем дело с УВВ 18F – конкретным объектом обмена информацией, именем тома "Информационная база", для обменных операций используется режим прямой адресации типа DA и каталог тома B¤( ) начинается с первого сектора диска.

Приводимый ниже пример иллюстрирует работу с каталогом файлов на томе при записи массивов на диск и их последующем считывании. Для автоматизации этих операций используются две упомянутые выше процедуры с формальными параметрами.

Вкратце пример сводится к следующему. Создается массив квадратов целых чисел и часть этого массива записывается на заданный диск в файл под указанным именем. После этого часть файла процедурой чтения считывается с диска обратно в ламять ЭВМ и выводится на печать. На печать выводится также новое содержание каталога тома, скорректированное записью в него новых файлов AGN и ASV.

Листинг фрагмента программы приведен на следующей странице.

```
0 REM ПРИМЕР РАБОТЫ С КАТАЛОГОМ ТОМА В РЕЖИМЕ (DA)
1 DIM B%(20)42,D%8,F%3,T%18,M(1000)
2 INPUT "ЗАДАТЬ ИМЕНА ТОМА НА 18F И ФАЙЛА",D%,F%
3 FOR K=1TO1000:M(K)=K^2:NEXT K:N=500
4 T%="КВАДРАТЫ ЧИСЕЛ (501":GOSUB ' 42(D%,F%,M(N),T%)
5 MAT M=ZER:IF P0=0THEN6:PRINT "ERR WITH CODE P0=";P0
6 N=10:GOSUB ' 47(D%,F%,M(N)):SELECT PRINT0C(164)
7 PRINT "M(10): ";M(1);M(2);M(3);M(4);M(5);M(6);M(7);M(8);M(9);M(10)
8 PRINT :PRINT "КАТАЛОГ ФАЙЛОВ НА ТОМЕ (ИНФОБАЗА)":PRINT
9 FOR K=1TO20:PRINT B%(K):IF STR(B%(K),3,3)=' 'THEN10:NEXT K
10 PRINT "КОНЕЦ РАБОТЫ С КАТАЛОГОМ ТОМА В РЕЖИМЕ (DA)":END
11 % @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@:DEFFN ' 42(D%,F%,M(N),T%)
12 DATA LOAD BA F(0)K%:IF STR(K%,1,8)=D%THEN14
13 PRINT "ИМЯ ТОМА ОТЛИЧНО ОТ ЗАДАННОГО":P0=2:GOTO 24
14 DATA LOAD BA F(1)B%():FOR K=1TO20
15 IF STR(B%(K),3,3)="FFF"THEN16:NEXT K:GOTO 19
16 STR(B%(K),3,3)=F%:STR(B%(K),22,18)=T%:CONVERT STR(B%(K),9,4)TOP
17 MAT REDIM M(N):DATA SAVE DA F%(P,L)M()
18 CONVERT (L-1)TOSTR(B%(K),16,4),(@@@@):GOTO 20
19 PRINT "КАТАЛОГ ЗАПОЛНЕН ДЛЯ ЗАПИСИ ФАЙЛОВ":P0=3:GOTO 24
20 IF K<20THEN21:P0=1:PRINT "КАТАЛОГ ЗАПОЛНЕН":GOTO 24
21 STR(B%(K+1),3,3)="FFF":STR(B%(K+1),16,4)="1000"
22 CONVERT LTOSTR(B%(K+1),9,4),(@@@@)
23 STR(B%(K+1),26,11)="EMPTY SPACE":P0=0
24 DATA SAVE DA F%(1,L)B%():RETURN :% ПРОЦЕДУРА ЗАПИСИ
25 DEFFN ' 47(D%,F%,M(N)):DATA LOAD BA F(0)K%
26 IF STR(K%,1,8)=D%THEN27:PRINT "ERR WITH CODE P0=2":P0=2:GOTO 32
27 DATA LOAD DA F(1)B%():FOR K=1TO20
28 IF STR(B%(K),3,3)=F%THEN30:NEXT K:PRINT "ФАЙЛА ";D%;" НЕТ"
29 P0=4:GOTO 32
30 CONVERT STR(B%(K),9,4)TOP:MAT REDIM M(N)
31 DATA LOAD DA F(P)M():P0=0
32 RETURN :% ПРОЦЕДУРА ЧТЕНИЯ С ДИСКОВОГО ТОМА

M(10): 1 4 9 16 25 36 49 64 81 100

КАТАЛОГ ФАЙЛОВ НА ТОМЕ (ИНФОБАЗА)

: S&C : 0000 : 0004 : ИМЯ И КАТАЛОГ ТОМА :
: XEN : 0005 : 0024 : КВАДРАТЫ ЧИСЕЛ(501 :
: FFF : 0025 : 1000 : EMPTY SPACE :

КОНЕЦ РАБОТЫ С КАТАЛОГОМ ТОМА В РЕЖИМЕ (DA)
```

Из приведенного примера следует, что уже такая простая организация ведения файлов на дисковых томах в режиме прямой адресации секторов позволяет существенно автоматизировать отслеживание файлового хозяйства задач пользователя.

Рассмотрим теперь примеры оформления программ с использованием наиболее интересных языковых средств БЕЙСИКа и представляющих интерес для практической работы на ЭВМ "Искра-226".

Программа определения адресов секторов на дисках предназначена для получения десятичных значений номеров секторов диска по их двухбайтовым представлениям. Входом в программу служат: ФАУ диска и его тип; номер сектора и номер первого байта, содержащие двухбайтовый адрес искомого сектора. На выходе программы формируются десятичный номер сектора и его HEX-представление на диске. Листинг программы имеет вид:

```
0 INPUT "РАБОТУ ПРОДОЛЖИТЬ(1-Д/2-Н)",D:IF.D=2THEN1:GOSUB 10
1 PRINT "КОНЕЦ ВЫЧИСЛЕНИЯ АДРЕСОВ СЕКТОРОВ":END :% КОНЕЦ
10 DIM A%(256)1,K%3,B%50,C%50,S%1,S1%1,A%1,B1%1,B2%1,B3%1,B4%1
```

```
20 INPUT "ЗАДАТЬ ДИСК(18F/18R/1CF/1CR)",K×
30 INPUT "ВВЕСТИ АДРЕС СЕКТОРА И НОМЕР БАЙТА",M,N
40 B×="70 SELECT DISK 18F:DATA LOAD BA F(M)A×()"
50 STR(B×,50,1)=HEX(85):C×=B×:STR(C×,16,3)=K×
60 STR(C×,33,1)=STR(K×,3):STR(C×,41,8)=":GOTO 90":LOAD C×70,70,70
80 A×=STR(A×(),N,1):GOSUB 120:B1=L:B2=P1:A×=STR(A×(),N+1,1)
90 GOSUB 120:B3=L:B4=P1:L=B1*16^3+B2*16^2+B3*16+B4
100 PRINT /0C,"НОМЕР СЕКТОРА=";L,"HEX(XXXX)=";
110 HEXPRINT /0C,STR(A×(),N,2):GOTO 0
120 M=VAL(A×):L=INT(M/16):P=M/16-L:P1=ROUND(P*16,0):RETURN

НОМЕР СЕКТОРА= 32         HEX(XXXX)=1080
```

Программа корректировки секторов диска позволяет на указанном дисковом томе корректировать заданное число байтов (не более 16) в заданном месте нужного сектора. Перед корректировкой содержимое корректируемых секторов отображается в целях контроля на дисплей и пользователь может либо корректировать выбранную информацию, либо пропустить ее. Корректировочная информация вводится побайтно в HEX-виде, т. е. посредством HEX-функции. Данная программа дает возможность в оперативном режиме корректировать информацию в любом месте заданного диска. Листинг программы имеет следующий вид:

```
10 PRINT HEX(030712):DIM X×3,A×(256)1,E1×4,E2×4
20 PRINT "PGM (CORRSECT): КОРРЕКТИРОВКА СЕКТОРОВ"
30 PRINT " (АВТОР: АЛАДЬЕВ В. - СКБ МПСМ ЭССР)  "
40 PRINT "########################################"
50 INPUT "ЗАДАТЬ ДИСК(18F/18R/1CF/1CR)",X×
60 INPUT "УКАЗАТЬ ЧИСЛО КОРРЕКТИРУЕМЫХ БАЙТОВ (=16",Z
70 INPUT "ЗАДАТЬ СЕКТОР И ПОЗИЦИЮ В НЕМ",S,M
80 IF X×="18R"THEN110:IF X×="18F"THEN140
90 IF X×="1CR"THEN160:IF X×="1CF"THEN170
100 PRINT "ERR: ОШИБКА А АДРЕСАЦИИ ДИСКА":GOTO 50
110 DATA LOAD BA R(S)A×():GOSUB 280
120 ON ERROR E1×,E2×GOSUB260
130 DATA SAVE BA R×(S)A×():GOTO 240
140 DATA LOAD BA F(S)A×():GOSUB 280
150 DATA SAVE BA F×(S)A×():GOTO 240
160 SELECT DISK1CR:GOTO 110
170 SELECT DISK1CF:GOTO 140
180 PRINT "КОРРЕКТИРУЮТСЯ ";Z;" БАЙТОВ СЕКТОРА"
190 FOR I=1TOZ:INPUT "ВВЕСТИ ОЧЕРЕДНОЙ HEX(XX)-БАЙТ",Q×
200 STR(P×,I,1)=STR(Q×,1,1)
210 PRINT "ВВЕДЕН БАЙТ С НОМЕРОМ ";I:NEXT I
220 HEXPRINT STR(P×,1,Z):PRINT STR(P×,1,Z)
230 STR(A×(),M,Z)=STR(P×,1,Z):RETURN
240 INPUT "РАБОТУ ПРОДОЛЖИТЬ(1-ДА/2-НЕТ)",W
250 IF W=1THEN10:PRINT "END  OF  JOB":END
260 REM ОБРАБОТКА ОСОБЫХ СИТУАЦИЙ :PRINT HEX(030712)
270 PRINT AT(10,20):PRINT "СНЯТЬ ЗАЩИТУ С ";X×:RETURN
280 HEXPRINT STR(A×(),M,Z):PRINT STR(A×(),M,Z)
290 INPUT "КОРРЕКТИРОВАТЬ ИНФОРМАЦИЮ(1-НЕТ/2-ДА)",W
300 ON WGOTO240:GOSUB 180:RETURN
```

Программа "STATUS" служит для изменения статуса файлов на дисках, состоящего в снятии или установке признаков защиты либо изменении признака ликвидируемости файла. В начале программа предлагает пользователю определить диск, содержащий каталог с нужным файлом, а затем и само имя файла. При нахождении требуемого файла программа информирует пользователя о типе файла и признаке его ликвидируемости. После этого от пользователя запрашива-

ется заказ на изменение статуса файла: защиты (1), ликвидируемости (2) или пропуска файла (3). Завершив изменение статуса файла, программа информирует об этом пользователя и предлагает повторить либо закончить с ней работу. В процессе своей работы программа выдает сообщения об ошибках (E RR) в работе с файлом и информационные сообщения (ATT), требующие внимания пользователя. Листинг программы имеет вид:

```
10 REM ПРОГРАММА (STATUS) * ALADYEV * СКБ МПСМ ЭССР:PRINT HEX(0312)
20 PRINT AT(2,20):PRINT "ПРОГРАММА ИЗМЕНЕНИЯ СТАТУСА ФАЙЛОВ":PRINT
30 DIM D%3,F%8,A%(256)1,B3%19,B2%26,A%1
40 INPUT "ОПРЕДЕЛИТЬ ДИСК С ФАЙЛОМ(18F/18R/1CF/1CR)",D%
50 B3%="70 SELECT DISK 18F":STR(B3%,19,1)=HEX(85)
60 STR(B3%,16,3)=D%:LOAD B3%70,70,70
80 S=0:GOSUB ' 47(S):N=1:GOSUB ' 42(N):G=A-1:S=0
90 INPUT "ОПРЕДЕЛИТЬ ИМЯ ТРЕБУЕМОГО ФАЙЛА",F%
100 GOSUB ' 47(S):FOR K=1TO249:IF STR(A%(),K,8)=F%THEN120:NEXT K
110 IF S)=GTHEN320:S=S+1:GOTO 100
120 IF STR(A%(),K-8,2)=HEX(1080)THEN150:IF STR(A%(),K-8,2)=HEX(1180)THEN160
130 IF STR(A%(),K-8,2)=HEX(1100)THEN170:IF STR(A%(),K-8,2)=HEX(1000)THEN180
140 PRINT "ERR A1: МЕТКА ФАЙЛА ";F%;" В УК ОШИБОЧНА":GOTO 300
150 PRINT "ATTN: FILE ";F%;" - IS PROGRAMM":G=1:GOTO 190
160 PRINT "ATTN: FILE ";F%;" - IS SCRATCH-PROGRAMM":G=2:GOTO 190
170 PRINT "ATTN: FILE ";F%;" - IS SCRATCH-DATA":G=3:GOTO 190
180 PRINT "ATTN: FILE ";F%;" - IS PROBLEM DATA":G=4
190 INPUT "ИЗМЕНИТЬ ЕГО ЗАЩИТУ(1);ЛИКВИДИРУЕМОСТЬ(2);НЕ МЕНЯТЬ(3)",T
200 IF T^3#G(=20RT^3#G)=0THEN220
210 PRINT "ERR A2: ФАЙЛ ДАННЫХ НЕ ЗАБИВАЕТСЯ ! - ПОВТОРИТЕ":GOTO 190
220 ON TGOTO330,240,300:PRINT "ERR A3: ЗАКАЗ НЕВЕРЕН":GOTO 190
240 BOOL 6(STR(A%(),K-8,1),HEX(01))
290 GOSUB ' 67(S):PRINT "SCRATCH-STATE OF FILE ";F%;" IS ALTERED"
300 INPUT "РАБОТУ СО (STATUS) ПРОДОЛЖИТЬ(1-ДА/2-НЕТ)",Q:ON GGOTO10,380
320 PRINT "ERR: ФАЙЛА С ТАКИМ ИМЕНЕМ В УК НЕТ - ПОВТОРИТЕ":GOTO 80
330 GOSUB ' 42(K-6):GOSUB ' 47(A):BOOL 6(STR(A%(),10,1),HEX(04))
370 GOSUB ' 67(A):PRINT "ЗАЩИТА ФАЙЛА ";F%;" ИЗМЕНЕНА":GOTO 300
380 PRINT "РАБОТА С ПРОГРАММОЙ (STATUS) ПОЛНОСТЬЮ ЗАКОНЧЕНА !!!":END
390 DEFFN ' 47(S):DATA LOAD BA T(S)A%():RETURN
420 DEFFN ' 67(S):DATA SAVE BA T%(S)A%():RETURN
470 DEFFN ' 42(N):A%=STR(A%(),N,1):GOSUB 500:B1=L:B2=P1:A%=STR(A%(),N+1,1)
480 GOSUB 500:B3=L:B4=P1:A=B1#16^3+B2#16^2+B3#16+B4:RETURN
500 M=VAL(A%):L=INT(M/16):P=M/16-L:P1=ROUND(P#16,0):RETURN
```

Следующий пример отражает обмен информацией с дисками по шести параметрам, смысл которых ясен из листинга программы, приведенного на следующей странице.

Задавая параметр операции обмена информацией, можно процедурой чтения или записи на заданный диск в режиме прямой адресации секторов (начиная с нужного сектора) заносить двухмерный массив данных заданной размерности. В процедуре указывается также адрес возврата после ее выполнения.

В процессе своей практической деятельности пользователю часто необходимо иметь информацию о содержимом тех или иных секторов диска. Простая программа, листинг которой приводится на следующей странице, предназначена для распечатки на принтере в символьном и HEX-виде содержимого заданных секторов указанного диска.

Процедура выборочного или полного копирования информации с диска на диск наиболее часто используется в практической работе с ПО и данными на ЭВМ любого типа. Программа,

```
10 REM ПРИМЕР ОФОРМЛЕНИЯ ПРОЦЕДУРЫ ОБМЕНА С ДИСКАМИ
20 REM ПАРАМЕТРЫ * V   - МЕТКА ВОЗВРАТА УПРАВЛЕНИЯ;
30 REM ----------* O%  - ТИП ОПЕРАЦИИ ОБМЕНА;
40 REM ----------* D%  - ПОЛНЫЙ АДРЕС ДИСКА;
50 REM ----------* S   - НОМЕР СЕКТОРА С ФАЙЛОМ;
60 REM ----------* B() - 2-МЕРНЫЙ МАССИВ ОБМЕНА;
70 REM ----------* M,N - РАЗМЕРНОСТЬ МАССИВА B().
90 DIM B(50,50),D%3,O%1,C(10,5),D(3,3):M=10:N=5
100 FOR K=1TO50:FOR L=1TO50:B(K,L)=K+L:NEXT L:NEXT K
110 GOSUB ' 42(1,"W","1CF",500,B(M,N)):MAT B=ZER
120 MAT B=ZER:GOSUB ' 42(2,"R","1CF",500,B(10,5))
140 MAT PRINT B:LET M,N=3:GOSUB ' 42(3,"R","1CF",500,B(M,N))
9000 MAT PRINT B:END :DEFFN ' 42(V,O%,D%,B(M,N)):MAT REDIM B(M,N)
9010 DIM B1%21,B2%38,B3%38,U%1:B1%="9070 SELECT DISK 18F"
9020 B2%="9100 DATA SAVE BA F% (S)B()":U%=STR(U%,3,1)
9030 B3%="9100 DATA LOAD BA F  (S)B()"
9040 STR(B1%,21,1),STR(B2%,38,1),STR(B3%,38,1)=HEX(85)
9050 STR(B1%,18,3)=D%:STR(B2%,19,1),STR(B3%,19,1)=U%
9060 STR(B2%,28,10),STR(B3%,28,10)="":GOTO 9120":LOAD B1%9070,9070,9070
9080 IF O%="R"THEN9090:LOAD B2%9100,9100,9100
9090 LOAD B3%9100,9100,9100
9120 ON ERROR E1%,E2%THEN9130:RETURN
9130 ON VGOTO120,140,9000:' ОБРАБОТКА ВОЗВРАТА УПРАВЛЕНИЯ


10 DIM A%(256)1,X%3,B%19:PRINT HEX(0312)
20 PRINT "PGM (PRINTSEC): SECTORS DISK'S PRINTING"
40 B%="80 SELECT  DISK 18F":STR(B%,19,1)=HEX(85)
50 INPUT "ЗАДАТЬ ДИСК(18F/18R/1CF/1CR)",X%:STR(B%,16,3)=X%
60 INPUT "ЗАДАТЬ НАЧАЛЬНЫЙ И КОНЕЧНЫЙ СЕКТОРА",S,F:LOAD B%80,80,80
90 FOR L=STOF:DATA LOAD BA T(L)A%():GOSUB 120:GOTO 170
120 SELECT PRINT0C(128):PRINT "SECTOR NUMBER ",L:PRINT
130 PRINT /05,HEX(0312):PRINT /05,"PRINT SECTOR ",L
140 HEXPRINT STR(A%(),1,128):PRINT STR(A%(),1,128)
150 HEXPRINT STR(A%(),129,128):PRINT STR(A%(),129,128)
160 PRINT :SELECT PRINT05(80):RETURN
170 NEXT L:INPUT "РАБОТУ ПРОДОЛЖИТЬ(1-ДА/2-НЕТ)",W
180 ON WGOTO10:PRINT "END  OF  JOB !":' КОНЕЦ ПРОГРАММЫ
```

листинг которой приводится ниже, позволяет копировать с диска на диск (копирование может вестись и в пределах одного диска) заданные секторы, при этом размещение их на принимающем (входном) диске может быть произвольным:

```
10 COM A%(256)1,D1%3,D2%3,B%253,M,N,Q,T:PRINT HEX(0312)
20 PRINT TAB(10),"(COPYDISK): КОПИРОВАНИЕ ДИСКОВ"
30 PRINT TAB(10),"АВТОР: АЛАДЬЕВ В.З. * СКБ МПСМ"
50 INPUT "ЗАДАТЬ ВЫХОДНОЙ ДИСК(18F/18R/1CF/1CR)",D1%
60 INPUT "ЗАДАТЬ ВХОДНОЙ ДИСК (18F/18R/1CF/1CR)",D2%
70 INPUT "С КАКОГО СЕКТОРА НА КАКОЙ И СКОЛЬКО",M,N,Q
80 INPUT "КОНТРОЛЬНУЮ ПЕЧАТЬ ДЕЛАТЬ(1-ДА/2-НЕТ)",T:D=1
85 IF T=2THEN90:PRINT "ВКЛЮЧИТЬ АЦПУ":STOP "'CONTINUE'"
90 B%="210 DATA LOAD BA R/18,(M)A%():GOTO 230"
100 STR(B%,40)="310 DATA SAVE BA R/18,(N,L)A%():GOTO 330"
110 STR(B%,81)="220 DATA LOAD BA F/18,(M)A%()"
120 STR(B%,111)="320 DATA SAVE BA F/18,(N,L)A%()"
130 STR(B%,39,1),STR(B%,80,1),STR(B%,110,1),STR(B%,142,1)=HEX(85)
140 STR(B%,20,2),STR(B%,100,2)=STR(D1%,1,2)
150 STR(B%,59,2),STR(B%,130,2)=STR(D2%,1,2):LOAD B%0,0,165
165 PRINT HEX(0312):PRINT "ИДЕТ КОПИРОВАНИЕ ДИСКА ";D1%;"-)";D2%
170 GOSUB ' 42(M):IF T=2THEN191
180 HEXPRINT /0C,STR(A%(),1,128):PRINT /0C,STR(A%(),1,128)
190 HEXPRINT /0C,STR(A%(),129,128):PRINT /0C,STR(A%(),129,128)
191 GOSUB ' 47(N):PRINT AT(20,20)
192 PRINT "ЗАПИСАНО ";D;" ОСТАЛОСЬ ";Q-D;" СЕКТОРОВ"
193 IF D=QTHEN194:D=D+1:M=M+1:N=N+1:GOTO 170
194 PRINT "КОПИРОВАНИЕ ПОЛНОСТЬЮ ЗАВЕРШЕНО !"
195 INPUT "РАБОТУ ПРОДОЛЖИТЬ(1-ДА/2-НЕТ)",L:ON LGOTO10:END
200 DEFFN ' 42(M):IF STR(D1%,3,1)="F"THEN220
230 ON ERROR P1%,P2%GOTO170:RETURN
300 DEFFN ' 47(N):ON ERROR E1%,E2%GOSUB340
305 IF STR(D2%,3,1)="F"THEN320
330 RETURN
340 REM ОБРАБОТКА ОСОБЫХ СИТУАЦИЙ:PRINT HEX(030712)
350 PRINT AT(10,20):PRINT "ATTN: СНЯТЬ ЗАЩИТУ С ";D2%:RETURN
```

При копировании секторов пользователь по желанию может параллельно выводить на печать их содержимое в символьном и HEX-видах. В данной программе наряду с использованием ряда интересных средств БЕЙСИКа демонстрируется возможность (строка 300) подавления отрицательных последствий выполнения оператора LOAD (строка 150) в случае применения его в помеченных подпрограммах.

В связи с наличием двух режимов работы с дисками (режима каталога файлов и режима прямой адресации секторов) в ряде случаев желательно иметь средства перехода от одного режима к другому. Переход из режима каталога файлов в режим прямой адресации не составляет труда, обратная же процедура гораздо сложнее. Программа, листинг которой приводится ниже, предназначена для каталогизации файлов, созданных на диске по операторам режима прямой адресации секторов типа DA:

```
0 REM КАТАЛОГИЗАЦИЯ ФАЙЛОВ, СОЗДАННЫХ В РЕЖИМЕ СЕКТОРОВ (DA)
1 DIM D%3,F%8,A%(256)1,A%1,B%2,S1%2,S2%2,B1%19,S3%2,S4%2
2 B1%="5  SELECT DISK 1CF":STR(B1%,19,1)=HEX(85)
3 INPUT "ОПРЕДЕЛИТЬ ДИСК С ФАЙЛОМ(18F/18R/1CF/1CR)",D%
4 STR(B1%,16,3)=D%:LOAD B1%5,5,5
6 INPUT "ЗАДАТЬ НАЧАЛЬНЫЙ СЕКТОР ФАЙЛА",N:T=N
7 INPUT "ПРИСВОИТЬ ИМЯ КАТАЛОГИЗИРУЕМОМУ ФАЙЛУ",F%
8 GOSUB ' 47(D%,N-1):S1%=B%:INIT (HEX(00))M%
9 GOSUB ' 47(D%,N):IF STR(A%(),1,1)=HEX(1C)THEN10:N=N+1:GOTO 9
10 F=N:U=N-T+1:S2%=B%:GOSUB ' 47(D%,T):STR(A%(),2,8)=F%:GOSUB ' 67(D%,T)
11 IF STR(A%(),1,1)=HEX(01)THEN13:STR(M%,1,2)=HEX(1000):GOTO 14
12 STR(M%,1,2)=HEX(1000):GOTO 14
13 STR(M%,1,2)=HEX(1080)
14 STR(M%,9,8)=F%:STR(M%,3,2)=S1%:STR(M%,5,2)=S2%
15 INIT (HEX(00))G%:GOSUB ' 47(D%,0):GOSUB ' 42(1):L=0
16 FOR K=1TO239:IF STR(A%(),K,16)=G%THEN18:NEXT K
17 IF L)A-1THEN24:L=L+1:GOSUB ' 47(D%,L):GOTO 16
18 STR(A%(),K,16)=M%:GOSUB ' 67(D%,L):GOSUB ' 47(D%,U-1):S4%=B%
19 GOSUB ' 47(D%,0):STR(A%(),3,2)=S2%:GOSUB ' 42(5):IF A)=FTHEN20:STR(A%(),5,2)=
S2%
20 STR(A%(),3,2)=S2%:GOSUB ' 67(D%,0)
21 GOSUB ' 47(D%,F):STR(A%(),2,2)=S4%:GOSUB ' 67(D%,F)
22 PRINT "ФАЙЛ ЗАКАТАЛОГИЗИРОВАН - ПРОВЕРИТЬ ЕГО !"
23 INPUT "РАБОТУ ПРОДОЛЖИТЬ(1-ДА/2-НЕТ)",G:ON GGOTO3:GOTO 9
24 PRINT "УКАЗАТЕЛЬ КАТАЛОГА ЗАПОЛНЕН - РЕОРГАНИЗОВАТЬ ТОМ":STOP :GOTO 9
25 DEFFN ' 47(D%,S):DATA LOAD BA T(S,B%)A%():IF ENDTHEN 10
26 RETURN :DEFFN ' 67(D%,S):ON ERROR E1%,E2%GOSUB31
27 DATA SAVE BA T%(S)A%():RETURN :DEFFN ' 42(N):A%=STR(A%(),N,1)
28 GOSUB 30:B1=V:B2=P1:A%=STR(A%(),N+1,1):GOSUB 30:B3=V:B4=P1
29 A=B1%16^3+B2%16^2+B3%16+B4:RETURN
30 N=VAL(A%):V=INT(N/16):P=N/16-V:P1=ROUND(P%16,0):RETURN
31 REM ОБРАБОТКА ОСОБЫХ СИТУАЦИЙ:PRINT HEX(030712)
32 PRINT AT(10,20):PRINT "ATTN:"СНЯТЬ ЗАЩИТУ С: ";D%:RETURN
```

Каталогизируемый файл должен размещаться на том же диске, что и УК тома, вне тела каталога или в самом каталоге. Аналогичным образом можно каталогизировать временные файлы данных, созданные в режиме каталога файлов.

В процессе каталогизации файла программой выполняются следующие основные процедуры. По входным данным определяются конечный сектор файла, содержащий метку конца HEX(IC), и тип файла (программа или данные), и на основе имеющейся информации формируется образ метки файла. После этого в УК тома отыскивается свободное место

для метки файла и в него помещается созданный образ метки. В последний сектор файла сразу за конечной меткой записывается вычисленное число реально занимаемых секторов под файл, а следующий свободный за файлом сектор диска заносится в УК вместо текущего свободного сектора каталога.

Если закаталогизированный файл находится в пределах каталога, то дальнейшая корректировка каталога не производится; в противном случае верхняя граница каталога сдвигается так, чтобы файл вошел в тело каталога. Программа информирует пользователя о результатах каталогизации и дает возможность повторить работу по каталогизации других файлов или завершить ее.

Вопрос эффективного размещения информации на дисках достаточно актуален и его значимость возрастает применительно к ЭВМ "Искра-226", устройства прямого доступа которой обладают сравнительно ограниченными возможностями. В качестве одного подхода в этом направлении рассмотрим достаточно простой способ упаковки символьной информации при записи ее на диск и последующей распаковки при считывании с диска в память.

Этот подход реализуется в операционной среде БЕЙСИКа, на основе помеченной процедуры DEFFN' 42 (A, M) с двумя формальными параметрами: A – тип процедуры и M – длина упаковываемого сообщения. Алгоритм реализации процедуры состоит в следующем.

Рабочими полями процедуры являются входной буфер G ¤( )длиной M, содержащий упаковываемое (распаковываемое) сообщение. Максимальная длина каждого буфера не превышает 25600 байтов, т. е. 100 секторов диска любого типа. Длина входного буфера задается пользователем, а выходного – вычисляется автоматически. Параметр A определяет тип процедуры (1 – упаковка, 2 – распаковка информации). В режиме упаковки процедурой кодируется символ кратности $к \geqslant 3$ ($a^к = aa...a$) в виде aHEX(OKp) X^p, где $p \in \{1,2,3\}$, $K_1=5, K_2=6$, $K_3=$ F и $X \in \{0-9\}$. Например, aHEX (06) 47 представляет символ a кратности 47.

Таким образом, при данной реализации процедуры кратность символа упаковываемой информации не должна превышать 999, что вполне достаточно для большинства задач пользователя.

В переменной U формируется код завершения процедуры: 1 – недостаточный размер выходного буфера; 2 – успешное завершение упаковки заданного сообщения; 3 – успешное завершение распаковки сообщения; 4 – неправильный тип процедуры.

Приведенный ниже листинг программы демонстрирует примеры использования процедуры упаковки (распаковки)

информации и результат ее выполнения. Само тело процедуры занимает ПС с номерами 5–30:

```
 0 % (PACK/UNPACK)-УПАКОВКА/РАСПАКОВКА СИМВОЛЬНОЙ ИНФОРМАЦИИ
 1 DIM G%(25600)1,S%(25600)1:INPUT "РАЗМЕР PACK-СООБЩЕНИЯ",N
 2 INPUT "ВВЕСТИ СООБЩЕНИЕ",STR(G%(),1,N):PRINT STR(G%(),1,N)
 3 A=1:GOSUB ' 42(A,N):GOSUB ' 47:INIT (' ')G%():A=2
 4 STR(G%(),1)=STR(S%(),1,LEN(S%())):INIT (' ')S%()
 5 GOSUB ' 42(A,N):GOSUB ' 47:END :DEFFN ' 42(A,N):MAT REDIM G%(N)
 6 DIM S%3:T,K,L=1:ON AGOTO7,16:U=4:GOTO 30
 7 G=1:GOTO 8:DEFFN ' 67(V,L):STR(S%(),V,1)=STR(G%(),L,1):RETURN
 8 L=L+1:IF STR(G%(),K,1)()STR(G%(),L,1)THEN9:G=G+1:IF L=NTHEN9:GOTO 8
 9 IF G(3THEN15:GOTO 10:DEFFN ' 0(C):DIM A%1:A%=STR(G%(),C+1,1):RETURN
10 STR(S%(),T,1)=STR(G%(),K,1):CONVERT GTOS%,(@@@):IF G(10THEN12
11 IF G(100THEN13:STR(S%(),T+1)=HEX(0F):STR(S%(),T+2)=S%:T=T+5:GOTO 14
12 STR(S%(),T+1)=HEX(05):STR(S%(),T+2)=STR(S%,3,1):T=T+3:GOTO 14
13 STR(S%(),T+1)=HEX(06):STR(S%(),T+2)=STR(S%,2,2):T=T+4:GOTO 14
14 K=K+G:L=K:IF K)NTHEN28:GOTO 7
15 STR(S%(),T)=STR(G%(),K,1):K=K+1:T=T+1:L=K:IF K(NTHEN7:U=2:GOTO 30
16 INIT (  ')S%():FOR K=1TOLEN(G%()):GOSUB 23:IF U=0THEN17:GOSUB 22
17 T=T+1:NEXT K:IF T)25600THEN27:MAT REDIM S%(T):V,L=1:GOSUB ' 0(L)
18 IF A%()HEX(05)ANDA%()HEX(06)ANDA%()HEX(0F)THEN21
19 K=L+1:GOSUB 23:CONVERT STR(G%(),L+2,U)TOM:FOR J=1TOM
20 GOSUB ' 67(V,L):V=V+1:NEXT J:L=L+U+2:IF L)LEN(G%())THEN29:GOTO 18
21 GOSUB ' 67(V,L):V=V+1:L=L+1:IF L)LEN(G%())THEN29:GOSUB ' 0(L):GOTO 18
22 CONVERT STR(G%(),K+1,U)TOM:T=T+M-1:RETURN
23 U=0:IF STR(G%(),K,1)()HEX(05)THEN24:U=1:GOTO 26
24 IF STR(G%(),K,1)()HEX(06)THEN25:U=2:GOTO 26
25 IF STR(G%(),K,1)()HEX(0F)THEN26:U=3
26 RETURN :DEFFN ' 47:PRINT :PRINT STR(S%(),1,LEN(S%())):RETURN
27 U=1:PRINT "ERR: РАЗМЕР ВЫХОДНОГО БУФЕРА ДОЛЖЕН БЫТЬ ) ";T:GOTO 30
28 U=2:PRINT "ATTN: СООБЩЕНИЕ УПАКОВАНО":GOTO 30
29 U=3:PRINT "ATTN: СООБЩЕНИЕ РАСПАКОВАНО"
30 RETURN :% КОНЕЦ ПРОЦЕДУРЫ (PACK/UNPACK)

RAA     GGGGGGGG      SSSSSKKKKKKKKKKVVVVVTTTTT
ATTN: СООБЩЕНИЕ УПАКОВАНО

RAA 4G8 5S5K10V5T5
ATTN: СООБЩЕНИЕ РАСПАКОВАНО

RAA     GGGGGGGG      SSSSSKKKKKKKKKKVVVVVTTTTT
```

Листинг рассмотренной процедуры ясен и особых пояснений не требует.

Программное обеспечение ЭВМ "Искра-226" содержит ряд средств для работы с текстовой информацией [30,36,43]. В качестве примера рассмотрим простую программу, позволяющую работать с текстами и не требующую специальной подготовки для работы с ней. В настоящее время эта программа, именуемая программой "Редактор", используется для документирования ПО, разрабатываемого в СКБ Министерства промышленности строительных материалов ЭССР, и подготовки различного ряда текстов. Ее листинг ясен и особых пояснений не требует:

```
 10 REM REDAKTOR 10.12.86 ALADYEV СКБ МПСМ ЭССР
 20 DIM A%(256)1,P%1,B%64,E%4,F%4:PRINT HEX(030712)
 30 PRINT "ПРОГРАММА РАБОТЫ С ТЕКСТАМИ © ALADYEV"
 40 PRINT "MOUNT DISK WITH TEXT IN 18F":STOP "CONTINUE"
 50 INPUT "ЗАДАТЬ НАЧАЛЬНЫЙ СЕКТОР ДЛЯ ТЕКСТА",P:L=P
 60 L=P:INPUT "PRINT(1);FORMATION(2);REDACTION(3);END(4)",Q
 70 ON ERROR E%,F%GOSUB450
 80 ON GOOTO470,90,340,330:GOTO 10:PRINT AT(1,1)
 90 PRINT HEX(030712):S=0
100 INPUT "ОПРЕДЕЛИТЬ (ИДЕНТИФИКАТОР) БОЛЬШИХ БУКВ",P%
110 PRINT HEX(0312):PRINT "BIG LETTERS (IDENTIFICATOR)=";P%
120 PRINT "ВВОДИТЬ СТРОКИ ТЕКСТА - '*'- ПРОПУСК СТРОКИ":S=S+1
130 PRINT "ПЕРЕД ЗАГЛАВНЫМИ БУКВАМИ НАЖАТЬ (ИДЕНТИФИКАТОР)"
140 FOR I=1TO256:STR(A%(),I,1)=HEX(20):NEXT I:PRINT AT(10,1)
150 PRINT AT(10,1):IF S=6THEN190:INIT (HEX(20))B%
```

```
160 LINPUT "ВВОД СТРОКИ:",-B$:GOSUB 570:S=S+1
170 IF STR(B$,1,1)()"*"THEN180:INIT (HEX(20))B$
180 STR(A%(),1+(S-2)*64,64)=B$:INIT (" ")B$:IF S<5THEN150
190 DATA SAVE BA F%(P)A%():P=P+1
200 INPUT "ПРОДОЛЖИТЬ ФОРМИРОВАТЬ ТЕКСТ(1-ДА/2-НЕТ)",Q:S=0
210 ON QGOTO110,220
220 STR(A%(),1,3)="FFF":DATA SAVE BA F%(P,T)A%()
230 PRINT "ТЕКСТ НА СЕКТОРАХ: "¡L¡" - "¡T-1:P=L:GOTO 60
240 SELECT PRINT0C(128):S=1:M=0:Z=0
250 INIT (HEX(20))A%():DATA LOAD BA F(P)A%():M=M+1
260 IF STR(A%(),1,3)="FFF"THEN320
270 IF M<=14THEN290:Z=Z+1:PRINT :PRINT TAB(35),"- "¡Z¡"
280 STOP "УСТАНОВИТЬ СТРАНИЦУ - 'CONTINUE'":M=0
290 PRINT TAB(10),STR(A%(),1+(S-1)*64,64):S=S+1
300 IF S=5THEN310:GOTO 290
310 P=P+1:S=1:GOTO 250
320 SELECT PRINT05(80):PRINT "ТЕКСТ ВЫВЕДЁН":P=L:GOTO 60
330 PRINT "КОНЕЦ РАБОТЫ С ПРОГРАММОЙ (TEXT)":END
340 INPUT "ЗАДАТЬ НОМЕР РЕДАКТИРУЕМОЙ СТРОКИ",N
350 IF N/4-INT(N/4))0THEN360:P=L+N/4-1:S=4:GOTO 380
360 P=L+INT(N/4)
370 DATA LOAD BA F(P)A%():S=(N/4-INT(N/4))*4
380 DATA LOAD BA F(P)A%()
390 B$=STR(A%(),1+(S-1)*64,64):PRINT HEX(030712)
400 PRINT AT(10,1):PRINT B$:PRINT AT(10,1)
410 LINPUT "РЕДАКТИРОВАНИЕ: ",B$
420 STR(A%(),1+(S-1)*64,64)=B$:DATA SAVE BA F%(P)A%()
430 P=L:INPUT "REDACTION DO(1-Y/2-N)",R:ON RGOTO340
440 PRINT HEX(0312):GOTO 60
450 PRINT HEX(030712):PRINT AT(10,20)
460 PRINT "СНЯТЬ ЗАЩИТУ С УВВ (18F)":RETURN
470 INPUT "ВЫВОД ТЕКСТА НА АЦПУ(1) ИЛИ ДИСПЛЕЙ(2)",W
480 ON WGOTO240:PRINT HEX(0312):S=1:M,N=0
490 INIT (HEX(20))A%():DATA LOAD BA F(P)A%()
500 M=M+1:IF STR(A%(),1,3)="FFF"THEN320
510 N=N+1:PRINT N;STR(A%(),1+(S-1)*64,64):S=S+1
520 IF S=5THEN530:GOTO 510
530 IF M/5-INT(M/5)=0THEN550
540 P=P+1:S=1:GOTO 490
550 STOP "ДЛЯ ПЕРЕЛИСТРАЦИИ ТЕКСТА - 'CONTINUE'"
560 PRINT HEX(0312):P=P+1:S=1:GOTO 490
570 DIM W$64,K%1:K,Q=1:U=0:K$=HEX(20):IF P$=HEX(00)THEN680
580 IF K)64THEN640:IF STR(B$,K,1)=P$THEN630
590 GOSUB   42(K):IF STR(B$,K,1)=HEX(40)THEN620
600 IF L1()14ANDL1()15ANDL1()4ANDL1()5THEN620
610 BOOL 6(STR(B$,K,1),K$)
620 K=K+1:GOTO 580
630 K=K+2:GOTO 580
640 IF Q)64THEN670:IF STR(B$,Q,1)=P$THEN660
650 U=U+1:STR(W$,U,1)=STR(B$,Q,1)
660 Q=Q+1:GOTO 640
670 B$=W$
680 RETURN :DEFFN   42(K):M1=VAL(STR(B$,K,1)):L1=INT(M1/16):RETURN
```

Программа обеспечивает следующее: 1) формирование и редактирование текста; 2) печать текста на принтер и дисплей. Длина строки формируемого текста составляет 64 символа в связи с тем, что сектор диска вмещает ровно четыре такие строки. Ввод текста осуществляется построчно в специально высвечиваемый на дисплее шаблон. В процессе ввода строка может редактироваться средствами языка БЕЙСИК (подобно режиму EDIT). Более того, при вводе текста пользователь получает возможность формировать текст, содержащий заголовочные и прописные буквы (латинские и русские). Для этого ему в ответ на запрос определить ⟨идентификатор⟩ следует ввести либо HEX(00) — ввод однородного текста, либо символ, отсутствующий в формируемом тексте. Тогда при необходимости выделения заголовочных букв перед ними нужно вводить данный идентификатор. Принцип работы с заго-

ловочными и прописными буквами подробно рассматривается ниже.

Введенные строки текста группами по четыре сразу же переписываются на диск. По окончании формирования текста пользователь получает возможность выполнить одну из вышеперечисленных процедур. В режиме редактирования он по номеру вызывает нужную строку текста и редактирует ее всеми средствами, используемыми в режиме EDIT. Отредактированная строка возвращается обратно на диск, и работу с программой можно продолжить.

При работе с текстовой информацией пользователь, по возможности, желал бы оперировать существующими правилами синтаксиса. ЭВМ "Искра-226" позволяет это делать, но здесь имеется ряд ограничений. Во-первых, алфавит не всех принтеров и дисплеев располагает заголовочными буквами (например, в АЦПУ "Роботрон-1154" их нет, а в АЦПУ ДЗМ-180 и Д-180 они есть). Во-вторых, непосредственный ввод с клавиатуры заголовочных букв затруднителен.

С другой стороны, HEX-представления заголовочных и прописных букв в операционной среде БЕЙСИКа различны. Поэтому есть основание для использования в текстовой информации заголовочных букв при наличии принтеров и дисплеев соответствующих типов. Ниже на простом примере описывается один подход к программной реализации такой возможности. В основу его положено следующее.

Между HEX-кодами заголовочных и прописных букв русского алфавита существуют простые соотношения

EX/FY ↔ CX/DY,

а между HEX-кодами латинского алфавита — аналогичные соотношения

4X/52 ↔ 6X/7Z,

где X ∈ {0–F}; Y ∈ X \ {F}; Z = {0–A}.

Нетрудно убедиться в том, что, определив символьную константу K¤ = HEX(20) и использовав булеву функцию языка БЕЙСИК BOOL 6(A¤,K¤), можно легко перейти от HEX-представления прописных букв к HEX-представлению соответствующих им заголовочных букв и наоборот.

С клавиатуры по операторам INPUT и LINPUT буквенная информация вводится в HEX-представлениях EX, FY, 4X и 5Z. Более того, если ЭВМ оснащена принтером, не допускающим букв обоих типов, то два типа HEX-представлений для буквенной информации совпадают по реакции на них принтера или дисплея. Поэтому напрашивается следующая простая схема программной реализации работы с заголовочными и прописными буквами в операционной среде БЕЙСИКа.

Символьная информация вводится с клавиатуры в символьную переменную A¤. При необходимости ввода заголовочной буквы перед ней набирается символ идентификатора, который не встречается в тексте. После заполнения переменной A¤ с помощью маски K¤ = HEX(20) и булевой функции BOOL 6(A¤, K¤) все HEX-представления буквенных символов A¤ преобразуются в HEX-представления соответствующих им прописных букв, за исключением тех букв, перед которыми стоят HEX-коды идентификатора заголовочных букв. Следовательно, на обычные принтеры подготовленный таким образом текст выводится однородными буквами, а на принтеры, допускающие два типа букв, – заголовочными и прописными буквами.

Ниже приводятся листинг фрагмента программы, реализующей описанный порядок работы с буквами двух типов, и результат его выполнения на примере конкретного текста.

```
10 REM РАБОТА С ЗАГЛАВНЫМИ БУКВАМИ:K,L=1:DIM A¤*253,B¤*253,K¤*1,P¤*1
20 PRINT HEX(0312):INPUT "ИДЕНТИФИКАТОР ЗАГЛАВНЫХ БУКВ",P¤
30 PRINT "ВВОДИТЬ ТЕКСТ;ПЕРЕД ЗАГЛАВНЫМИ - (ИДЕНТИФИКАТОР)"
40 LINPUT "ВВОД СТРОКИ ТЕКСТА",A¤:K¤=HEX(20)
50 IF K)253THEN100:IF STR(A¤,K,1)=P¤THEN90
60 GOSUB ' 42(K):IF STR(A¤,K,1)=HEX(40)THEN80
70 IF L()14ANDL()15ANDL()4ANDL()5THEN80:BOOL 6(STR(A¤,K,1),K¤)
80 K=K+1:GOTO 50:REM ALADYEV @ SKB MPSM ESSR
90 K=K+2:GOTO 50:REM WAS CREATED IN APRIL 1986
100 IF L)253THEN130:IF STR(A¤,L,1)=P¤THEN120
110 P=P+1:STR(B¤,P,1)=STR(A¤,L,1)
120 L=L+1:GOTO 100
130 SELECT PRINT@C(50):PRINT B¤:SELECT PRINT@5(80):GOTO 150
140 DEFFN ' 42(K):M=VAL(STR(A¤,K,1)):L=INT(M/16):RETURN
150 INPUT "ПОВТОРИТЬ(1-ДА/2-НЕТ)",D:IF D=1THEN10:END

Данный Текст Является Тестовым . КОНЕЦ !
```

Заметим, что формирование текста рассмотренным способом требует большего расхода машинного времени (из-за побайтной перекодировки HEX-представлений символов вводимых строк текста), но при выводе сформированного текста на печать перерасхода машинного времени не будет.

Второй подход к организации работы с буквами двух типов более прост, требует меньше времени на формирование текста и в нем используется одно интересное свойство операторов ввода информации с клавиатуры. Посредством всех операторов ввода с клавиатуры, кроме оператора KEYIN, информация вводится в верхнем регистре (заголовочные буквы), тогда как с помощью оператора KEYIN – в нижнем регистре (прописные буквы). А так как большую часть текста составляют прописные буквы, то процесс формирования текста очень прост. Основной ввод реализует оператор KEYIN , формируя прописные буквы текста. При необходимости ввода заголовочной буквы нажимается любая клавиша КСФ и заголовочная буква вводится по оператору INPUT.

Следовательно, оператор KEYIN в данном случае не только используется как основной оператор ввода информации

с клавиатуры, но и служит для организации управления вводом того или иного типа букв как латинского, так и русского алфавитов.

Изложенное иллюстрирует листинг программы:

```
0 REM РАБОТА С ЗАГЛАВНЫМИ БУКВАМИ:DIM A%50,K%1:T=0
1 PRINT HEX(0312):PRINT "ВВОД СТРОКИ В 50 СИМВОЛОВ"
2 PRINT "ПЕРЕД ЗАГЛАВНЫМИ НАЖИМАТЬ ЛЮБУЮ (КС#)"
3 KEYIN K%,4,5:GOTO 3
4 IF K%=HEX(85)THEN6:IF T>50THEN6:T=T+1:GOSUB 9:GOTO 3
5 IF T>50THEN6:T=T+1:INPUT "ЗАГЛАВНАЯ",K%:GOSUB 9:GOTO 3
6 PRINT /0C,A%:INPUT "ПРОДОЛЖИТЬ(1-ДА/2-НЕТ)",L
7 IF L=2THEN8:T=0:INIT (HEX(20))A%:GOTO 1
8 PRINT :PRINT /0C,"(@) ИСКРА 226":END
9 PRINT HEX(0312):STR(A%,T,1)=K%:PRINT AT(10,10):PRINT A%:RETURN

Работа с Заглавными и Прописными Буквами - ТЕКСТ
(@) ИСКРА 226
```

Согласно этой программе пользователю предлагается формировать строки по 50 символов, содержащие буквы любого из двух типов. После завершения формирования строки (нажатия клавиши C R/LF или заполнения строки) она выводится на печать. Рассмотренный способ формирования текста легко адаптировать к потребностям любой программы пользователя, работающего с текстовой информацией.

В ряде случаев возникает необходимость такого редактирования текстовой информации, когда все буквенные символы должны быть сделаны заголовочными при выводе на принтер любого типа. Эту процедуру реализует программа, входными данными для которой являются адрес диска, начальный сектор и число секторов, содержащих корректируемый текст. Ее листинг имеет вид:

```
10 REM РЕДАКТИРОВАНИЕ ТЕКСТА НА ДИСКА * ALADYEV * 17.12.86
20 DIM A%(256)1,K%1,D%3,B%20:K%=HEX(20):B%="60 SELECT DISKXXX"
30 INPUT "ОПРЕДЕЛИТЬ ДИСК(18F/18R/1CF/1CR)",D%
40 INPUT "ЗАДАТЬ НАЧАЛЬНЫЙ СЕКТОР И ЧИСЛО СЕКТОРОВ",M,D:V=M
50 STR(B%,15,3)=D%:STR(B%,20,1)=HEX(85):LOAD B%60,60,60
60 SELECT DISK18R
70 DATA LOAD BA T(M)A%():FOR K=1TO256
80 GOSUB ' 42(K):IF STR(A%(),K,1)=HEX(40)THEN110
90 IF L1()12ANDL1()13ANDL1()6ANDL1()7THEN110:GOTO 100
100 BOOL 6(STR(A%(),K,1),K%)
110 NEXT K:GOTO 130
120 DEFFN ' 42(K):M1=VAL(STR(A%(),K,1)):L1=INT(M1/16):RETURN
130 DATA SAVE BA T(M)A%():M=M+1:IF M(D+VTHEN70:PRINT HEX(0312)
140 PRINT AT(10,10):PRINT "ОТРЕДАКТИРОВАНЫ СЕКТОРА ";V;"-";M-1
```

Наконец, программа, листинг которой приводится ниже, дает возможность редактировать уже имеющийся текст на диске:

```
20 DIM A%(256)1,K%1,D%3,B%64,C%64:B%="60SELECTDISKXXX"
30 INPUT "ОПРЕДЕЛИТЬ ДИСК(18F/18R/1CF/1CR)",D%
40 INPUT "ЗАДАТЬ НАЧАЛЬНЫЙ И КОНЕЧНЫЙ СЕКТОРА",M,D
50 STR(B%,16,1)=HEX(85):STR(B%,13,3)=D%:LOAD B%60,60,60
70 FOR J=MTOD:DATA LOAD BA T(J)A%():FOR K=1TO192STEP64
80 B%=STR(A%(),K,64):PRINT HEX(0312):INIT (HEX(00))C%
90 PRINT ,"В БАБЛОНЕ ПОД НЕИЗМЕНЯЕМЫМ СИМВОЛОМ НАДАТЬ "
100 PRINT ,"КЛАВИШУ '->',ИЗМЕНЯЕМЫМ СИМВОЛОМ - ПРОБЕЛ-' '"
110 PRINT AT(15,0):PRINT B%:PRINT AT(16,1):LINPUT "БАБЛОН:",-C%
120 FOR L=1TO64:BOOL 6(STR(B%,L,1),STR(C%,L,1))
130 NEXT L:INIT (HEX(00))C%:STR(A%(),K,64)=B%:NEXT K
140 DATA SAVE BA T(J)A%():NEXT J:END :% КОНЕЦ РЕДАКТИРОВАНИЯ
```

6 8-310

Входными данными для нее являются адрес диска с текстом, начальный и конечный секторы диска. Редактирование текста осуществляется построчно частями по 64 символа следующим образом. Очередная часть редактируемого текста выводится на дисплей и строго под ней высвечивается корректировочный шаблон на 64 символа. Под символами части программы, которые следует оставить без изменения, нажимается клавиша -->, используемая в режиме EDIT, а под изменяемыми символами в шаблоне – клавиша пробела. Поскольку шаблон формируется с помощью оператора LINPUT, информацию в нем до нажатия клавиши CR/LF можно корректировать всеми средствами, используемыми в режиме EDIT.

Рассмотрение организации работы с прописными и строчными буквами в операционной среде БЕЙСИКа завершим еще одним интересным примером. Как уже не раз отмечалось, буквенная информация и только она, представленная в апострофах, вводится в ЭВМ посредством оператора INPUT в нижнем регистре (шестнадцатиричное представление которого соответствует прописным буквам), а представленная в кавычках или без них – в верхнем регистре (шестнадцатиричное представление которого соответствует заголовочным буквам).

Данное обстоятельство может быть положено в основу создания программ для обработки буквенной информации двух типов: заголовочных и прописных букв как латинского, так и русского алфавитов. Приведенный ниже листинг программы демонстрирует использование описанного подхода для организации работы с буквами двух типов: заголовочных и прописных как латинского, так и русского алфавитов.

Приведенный ниже листинг программы демонстрирует использование данного подхода для организации работы с буквами двух типов:

```
0 INPUT "ЗАДАТЬ ДИСК И ИМЯ ФАЙЛА С ТЕКСТОМ",Dx,Fx
1 Tx="3 SELECTDISKTTT":STR(Tx,16,1)=HEX(8S):STR(Tx,13,3)=Dx
2 LOAD Tx3,3,3:% ФОРМИРОВАНИЕ АДРЕСА РЕЗИДЕНТНОГО ДИСКА
3 SELECT DISK1CR
4 DIM Ax64,Bx64:INIT (20)Ax,Bx:% ОРГАНИЗАЦИЯ РАБОТЫ С БУКВАМИ ДВУХ ТИПОВ
5 INPUT "ВВОД ПРОПИСНЫХ БУКВ - В АПОСТРОФАХ(')",Bx:GOSUB 19:GOTO 6
6 INPUT "ВВОД ЗАГЛАВНЫХ БУКВ - В КАВЫЧКАХ ИЛИ БЕЗ НИХ",Bx:GOSUB 19:GOTO 5
7 PRINT "КОНЕЦ ФОРМИРОВАНИЯ ТЕКСТА С ДВУМЯ ТИПАМИ БУКВ":IF H=5THEN11
8 INPUT "ВЫВЕСТИ ТЕКСТ НА ПЕЧАТЬ(1) ИЛИ ДИСК(2)",P:ON PGOTO14
9 PRINT "ЗАПИСЬ СФОРМИРОВАННОГО ТЕКСТА ИДЕТ В КАТАЛОГ ТОМА (1CR)"
10 DATA SAVE DC OPEN T(200)Fx
11 DATA SAVE DC #0,Ax:IF Bx="FFF"THEN12:H=5:GOTO 4
12 DATA SAVE DC #0,END:DATA SAVE DC CLOSE #0
13 PRINT "СФОРМИРОВАННЫЙ ТЕКСТ ЗАПИСАН НА ТОМЕ ";Dx:GOTO 15
14 PRINT "ПОДГОТОВИТЬ ПРИНТЕР - НАЖАТЬ 'CONTINUE'":PRINT /0C,Ax:END
15 INPUT "ТЕКСТ С ДИСКА ВЫВОДИТЬ НА ПЕЧАТЬ(1-Д/2-Н)",T:ON TGOTO16:END
16 DATA LOAD DC OPEN TFx
17 DATA LOAD DC #0,Ax:IF ENDTHEN 18:PRINT /0C,Ax:GOTO 17
18 % КОНЕЦ РАСПЕЧАТКИ ТЕКСТА:DATA SAVE DC CLOSE #0:END
19 IF Bx=HEX(202020)THEN21:IF Bx="FFF"THEN7:L1=LEN(Ax)
20 L2=LEN(Bx):IF L1+L2)64THEN21:STR(Ax,L1+1,L2)=Bx
21 INIT (20)Bx:RETURN :% КОНЕЦ ФРАГМЕНТА ПРОГРАММЫ

Данный текст является тестовым для ПРОГРАММЫ
```

Согласно этой программе пользователь для ввода текстовых строк длиной не более 64 символов использует оператор INPUT. При этом заголовочные буквы вводятся в кавычках или без них, а прописные – в апострофах. Программа об этом информирует пользователя. Концом формирования текста служит набор символов FFF.

Сформированный текст можно выводить на печать или записывать на заданный диск с указанным именем. Листинг программы и результат ее выполнения ясны и пояснений не требуют.

В языке БЕЙСИК имеются средства для вывода содержимого каталога томов прямого доступа, речь о которых шла выше. Однако в ряде случаев эти средства не совсем удобны. Листинг программы для печати содержимого каталога тома имеет вид:

```
10 PRINT :PRINT :%     ПЕЧАТЬ СОДЕРЖИМОГО КАТАЛОГА ТОМА
20 PRINT :% (ПЕРЕЧЕНЬ ИСПОЛЬЗУЕМЫХ СОКРАЩЕНИЙ):PRINT
30 % :PRINT HEX(12):%                    :PRINT HEX(12)
40 PRINT :PRINT :% ПРОГРАММНЫЙ ФАЙЛ ПОЛЬЗОВАТЕЛЯ(ПФ)
50 PRINT :PRINT :PRINT :% ФАЙЛ ДАННЫХ ПОЛЬЗОВАТЕЛЯ(ФД)
60 PRINT :% УДАЛЕННЫЙ ПРОГРАММНЫЙ ФАЙЛ(ПО SCRATCH)(УПФ)
70 PRINT :% УДАЛЕННЫЙ ФАЙЛ ДАННЫХ(ПО SCRATCH)(УФД)
80 PRINT :% ТРАНСЛИРОВАННЫЙ ПРОГРАММНЫЙ ФАЙЛ(ТПФ)
90 PRINT :PRINT :% ЗАЩИЩЕННЫЙ ПРОГРАММНЫЙ ФАЙЛ(ЗПФ)
100 PRINT :% ЗАЩИЩЕННЫЙ И ТРАНСЛИРОВАННЫЙ ПФ(ЗТПФ)
110 DIM T%3,A%(256)1,B%20,F%8,Y%46,U%4,W%4,S%4,D%2
120 INPUT "ЗАДАТЬ ДАТУ В ВИДЕ (ДД.ММ.ГГГГ)",Z%
130 INPUT "ЗАДАТЬ ТОМ С КАТАЛОГОМ(18F/18R/1CF/1CR)",T%
140 INPUT "ВЫВОД КАТАЛОГА НА АЦПУ(1)/ДИСПЛЕЙ(2)",Q
150 ON Q-1GOTO160:SELECT PRINT@C(164):PRINT :PRINT
160 PRINT HEX(031207),:PRINTUSING 10:PRINT ,:PRINTUSING 20
170 PRINTUSING 30:PRINTUSING 40:PRINTUSING 50:PRINTUSING 60
180 PRINTUSING 70:PRINTUSING 80:PRINTUSING 90:PRINTUSING 100
190 STOP "CONTINUE":PRINT HEX(0312):B%="220 SELECT DISK XXX":K,Z=0
210 STR(B%,20,1)=HEX(85):STR(B%,17,3)=T%:LOAD B%220,220,220
220 SELECT DISK18F
230 DATA LOAD BA T(Z)A%():GOSUB ' 42(1):M0=A:GOSUB ' 42(3)
240 A1=A:GOSUB ' 42(5):PRINT ,"КАТАЛОГ ФАЙЛОВ НА ЗАДАННОМ ТОМЕ: ";T%
250 PRINT :PRINT ,"РАЗМЕР УКАЗАТЕЛЯ КАТАЛОГА НА ТОМЕ=";M0
260 PRINT ,"ТЕКУЩИЙ КОНЕЦ ТЕЛА КАТАЛОГА  ТОМА=";A1
270 PRINT ,"ФИКСИРОВАННЫЙ КОНЕЦ ТЕЛА КАТАЛОГА=";A
280 PRINT ,"СВОБОДНОЕ МЕСТО В КАТАЛОГЕ   ТОМА=";A-A1:PRINT
290 INIT (".")Y%:PRINT ,Y%:GOSUB 570:STR(Y%,3,3)="П/П"
300 STR(Y%,34,5)="ЗА-ТО":STR(Y%,41,4)="ТИП":STR(Y%,9,9)="ИМЯ ФАЙЛА"
310 STR(Y%,20,4)="НАЧ.":STR(Y%,27,4)="КОН.":PRINT ,Y%:GOSUB 570
320 STR(Y%,9,8)="КАТАЛОГА":STR(Y%,20,4),STR(Y%,27,4)="СЕКТ"
330 STR(Y%,34,4)="СЕКТ":STR(Y%,41,5)="ФАЙЛА":PRINT ,Y%
340 INIT (".")Y%:PRINT ,Y%:W=17:GOSUB ' 87(W):GOSUB 430
350 GOSUB ' 87(W):GOSUB 430:GOTO 350
360 INIT (HEX(2E))Y%:PRINT ,Y%:PRINT :GOSUB ' 20
370 PRINT ,"ЧИСЛО СВОБОДНЫХ МЕТОК ФАЙЛОВ В УК ТОМА=";G0:PRINT
380 PRINT ,"СОСТОЯНИЕ КАТАЛОГА ТОМА НА ";STR(Z%,1,10)
390 SELECT PRINT05(80):END :DEFFN ' 42(N):A%=STR(A%(),N,1)
400 GOSUB 410:B=L:C=P:A%=STR(A%(),N+1,1):GOSUB 410:D=L:E=P
410 A=B*16^3+C*16^2+D*16+E:RETURN
420 M=VAL(A%):L=INT(M/16):P=ROUND((M/16-L)*16,0):RETURN
430 DATA LOAD BA T(Z)A%():RETURN :DEFFN ' 86(F%,K)
440 LIMITS TF%,U,R,S,O:CONVERT UTOU%,(@@@@)
450 CONVERT RTOU%,(@@@@):CONVERT STOS%,(@@@@)
460 GOSUB 570:CONVERT KTOSTR(Y%,3,3),(@@@)
470 STR(Y%,9,8)=F%:STR(Y%,20,4)=U%:STR(Y%,27,4)=W%
480 STR(Y%,34,4)=S%:ON OGOTO490,550,500,510,520,530,540
490 STR(Y%,41,4)="ПФ":GOTO 560:REM ПОЛНАЯ РАСПЕЧАТКА КАТАЛОГА
500 STR(Y%,41,4)="УПФ":GOTO 560:REM ЗАДАННОГО ДИСКОВОГО ТОМА
510 STR(Y%,41,4)="УФД":GOTO 560:REM АВТОР * ALADYEV * 1986
520 STR(Y%,41,4)="ТПФ":GOTO 560:REM  SKB  MPSM  ESSR
530 STR(Y%,41,4)="ЗПФ":GOTO 560:% ТАЛЛИН 200006
540 STR(Y%,41,4)="ЗТПФ":GOTO 560:% МУСТАМЯЭ ТЕЕ 55
550 STR(Y%,41,4)="ФД":GOTO 560:% ТЕЛ. 527-665(4)
```

```
560 PRINT ,Y%:RETURN :DEFFN ' 20:G0=M0%16-K-1:RETURN
570 INIT (HEX(20))Y%:STR(Y%,1,1),STR(Y%,7,1),STR(Y%,18,1)=":"
580 FOR J=18TO46STEP7:STR(Y%,J,1)=":":NEXT J:RETURN
590 DEFFN ' 87(W):FOR I=WTO239STEP16:D%=STR(A%(),I,2)
600 IF D%()HEX(1080)ANDD%()HEX(1180)ANDD%()HEX(1100)ANDD%()HEX(1000)THEN620
610 K=K+1:F%=STR(A%(),I+8,8):GOSUB ' 86(F%,K)
620 NEXT I:X=X+1:IF X)=M0THEN360:W=1:Z=Z+1:RETURN
```

Входными данными для этой программы являются дата, адрес тома с каталогами и адрес вывода информации о каталоге (ФАУ-ОС или ФАУ-05). Информация, выдаваемая программой о файлах каталога, намного полнее, чем получаемая по оператору LIST DC.

Сортировка файлов – одна из наиболее часто используемых процедур обработки информации. Выше уже отмечалось, что средства БЕЙСИКа дают возможность осуществлять сортировку только в памяти ЭВМ, но они не полностью удовлетворяют сортировке файлов, расположенных на внешних носителях: пользователь должен либо сам программировать процедуры сортировки, либо использовать имеющиеся программные средства.

Рассмотрим пример организации процедуры сортировки символьных файлов на дисках. Процедура сортировки описывается восьмью формальными параметрами, содержание которых раскрыто в комментариях листинга процедуры:

```
10 REM ПРОЦЕДУРА СОРТИРОВКИ ФАЙЛОВ НА ДИСКАХ:DIM V%4,L%3
20 REM ----- D% - АДРЕС ДИСКА(18F/18R/1CF/1CR)
30 REM ----- F% - ИМЯ СОРТИРУЕМОГО ФАЙЛА
40 REM ----- V% - НОМЕР СТРОКИ ВОЗВРАТА(NNNN)
50 REM ----- L  - ДЛИНА ЛОГИЧЕСКОЙ ЗАПИСИ
60 REM ----- M,P- ПОЗИЦИИ КЛЮЧЕЙ СОРТИРОВКИ
70 REM ----- N,R- ДЛИНЫ КЛЮЧЕЙ СОРТИРОВКИ ФАЙЛА
80 PRINT "@@@@@@@@@@@@@@@@":COM M,N,P,R,D%3,F%8,K1,K2:GOSUB 500
90 M,N=1:R=2:P=8:V%="0094":L%="016":GOSUB ' 64(D%,F%,L%,M,N,P,R,V%)
94 PRINT /0C,"ОТСОРТИРОВАННЫЙ ФАЙЛ ";F%:PRINT /0C:SELECT @31C
95 DATA LOAD DC OPEN R@3,F%
96 DATA LOAD DC @3,W%:PRINT /0C,W%:IF ENDTHEN 98:GOTO 96
98 PRINT /0C,"КОНЕЦ РАБОТЫ ФРАГМЕНТА СОРТИРОВКИ ":END
99 PRINT "########################################"
100 DEFFN ' 64(D%,F%,L%,M,N,P,R,V%):DIM B5%30,C%21
101 C%="0104 DIM A%XXX,B%XXX":STR(C%,21,1)=HEX(85)
102 STR(C%,12,3),STR(C%,18,3)=L%:LOAD C%104,104,103
103 B5%="0415 ON ERROR K%,P% THEN XXXX":STR(B5%,26,4)=V%
105 DIM B1%30,B2%26,B3%26,B6%17,B4%22:INPUT #K1
106 B6%="0190 SELECT @51C":STR(B5%,30,1)=HEX(85)
110 B1%="0200 DATA LOAD DC OPEN F@5,F%"
120 B2%="0300 DATA LOAD DC @5,A%":B4%="0420 RETURN:GOTO XXXX"
130 B3%="0360 DATA SAVE DC @5,A%":STR(B1%,30,1)=HEX(85)
140 STR(B2%,26,1),STR(B4%,22,1),STR(B3%,26,1),STR(B6%,17,1)=HEX(85)
150 STR(B6%,15,2)=STR(D%,1,2):STR(B1%,24,1)=STR(D%,3,1)
160 STR(B4%,18,4)=V%:LOAD B6%190,190,200
200 LOAD B1%200,200,210
210 LOAD B4%210,210,220
220 LOAD B2%220,220,230
230 LOAD B3%230,230,235
235 LOAD B5%235,235,240
240 STR(B2%,1,4)="0310":STR(B2%,22,1)="B"
250 LOAD B2%250,250,260
260 STR(B3%,1,4)="0350":STR(B3%,22,1)="B"
270 LOAD B3%270,270,190
305 IF ENDTHEN 400
315 IF ENDTHEN 400
320 IF STR(A%,M,N))STR(B%,M,N)ANDSTR(A%,P,R))STR(B%,P,R)THEN340
330 DBACKSPACE @5,1:GOTO 300
340 DBACKSPACE @5,2:S0=424767
365 GOTO 300
```

```
400 IF S0=0THEN405:S0=0:DBACKSPACE #5,BEG:GOTO 300
405 DATA SAVE DC CLOSE #5
408 INPUT #K2:PRINT /0C,"ВРЕМЯ СОРТИРОВКИ ФАЙЛА ";F#;
409 PRINT /0C,"=";ROUND((K1-K2)/120000,2);" МИН"
410 PRINT "ФАЙЛ ОТСОРТИРОВАН НА ДИСКЕ ";D#
420 RETURN :GOTO 94
500 INPUT "ЗАДАТЬ ИМЯ СОРТИРУЕМОГО ФАЙЛА",F#
510 INPUT "ЗАДАТЬ ИМЯ ДИСКА",D#:SELECT #11C:DATA SAVE DC OPEN R#1,(42)F#
520 PRINT "ВВОДИТЬ СТРОКИ СИМВОЛОВ (=16 Б ; '*'- ПРОПУСК"
530 LINPUT "ВВОД ОЧЕРЕДНОЙ СТРОКИ СИМВОЛОВ",W#:IF STR(W#,1,1)='*'THEN580
540 PRINT HEX(0312):DATA SAVE DC #1,W#:INIT (HEX(20))W#:GOTO 530
570 DATA SAVE DC #1,END:DATA SAVE DC CLOSE #1:SELECT #11C
580 DATA LOAD DC OPEN R#1,F#
590 DATA LOAD DC #1,W#:PRINT /0C,W#:IF ENDTHEN 610:GOTO 590
600 IF ENDTHEN 610:GOTO 590
610 DATA SAVE DC CLOSE #1:RETURN
```

```
9999999999999999                    1111111111111111
EEEEEEEEEEEEEEEE                    3333333333333333
7777777777777777                    5555555555555555
DDDDDDDDDDDDDDDD                    7777777777777777
5555555555555555                    9999999999999999
CCCCCCCCCCCCCCCC                    AAAAAAAAAAAAAAAA
3333333333333333                    BBBBBBBBBBBBBBBB
BBBBBBBBBBBBBBBB                    CCCCCCCCCCCCCCCC
1111111111111111                    DDDDDDDDDDDDDDDD
AAAAAAAAAAAAAAAA                    EEEEEEEEEEEEEEEE

ВРЕМЯ СОРТИРОВКИ ФАЙЛА XENIX14 = 0.42 МИН
ОТСОРТИРОВАННЫЙ ФАЙЛ: XENIX14      КОНЕЦ СОРТИРОВКИ
```

В процедуре сортировки, реализованной на основе помеченной подпрограммы (строки 100–420) и в соответствии со значениями формальных параметров, посредством операторов LOAD генерируется свой текст для выполнения того или иного режима сортировки. Процедура позволяет сортировать символьные файлы на дисках любого типа. Длина записей файла – не более 253 байтов. В качестве ключей сортировки выбираются два поля с заданными их положением в записи и длиной. Для сортировки в процедуре реализуется пузырьковый метод, который не требует больших затрат ОП, а сам файл сортируется на месте, т. е. дополнительное место на внешних носителях информации не требуется.

Согласно приведенной программе на диске создается контрольный символьный файл, который выводится на печать. Затем процедурой сортировки сортируется файл и также выводится на печать. Время, затраченное на сортировку, хронометрируется.

В качестве контрольного примера была проведена сортировка символьного файла из 10 записей длиной 16 байтов по двум ключам длиной 1 и 2 байта. Эмпирический анализ времени этой сортировки показал, что для сортировки подобных файлов на жестких дисках требуемое максимальное время

$$T(N = 12 \times K) \sim 0{,}5 \cdot 2{,}35^{K-1},$$

где N – число сортируемых записей; к = 1,2,3,... .

Из этого анализа следует, что описанная процедура сортировки довольно медленная, хотя в ряде случаев она (или ей

подобные процедуры) может быть вполне приемлемой. Используя другие, более быстрые алгоритмы сортировки, пользователь может разработать свои собственные процедуры. При этом он может воспользоваться приведенными здесь примерами организации подобного рода процедур широкого назначения.

При работе с каталогизированными файлами и самим каталогом тома языковые средства БЕЙСИКа в ряде случаев не совсем эффективны, а порой и недостаточны. Поэтому была разработана программа SVEGAL, позволяющая осуществлять полную реорганизацию каталога дискового тома. Основные функции ее следующие:

физическое удаление файлов с последующим уплотнением тела каталога и соответствующей корректировкой УК;

увеличение УК на заданное число секторов с соответствующей реорганизацией тела каталога;

увеличение текущего и (или) фиксированного концов каталога тома;

вывод на дисплей или принтер полного состояния каталога тома с выдачей затраченного на работу времени.

Входными данными программы являются полный адрес дискового тома и номер нужного действия. Работа с программой осуществляется в диалоговом режиме по инициативе программы. Реорганизация каталога тома идет в рамках заданного тома, не требуя других дисковых томов. Листинг программы имеет следующий вид:

```
10 REM (SVEGAL)-ПОЛНАЯ МОДИФИКАЦИЯ КАТАЛОГА * АЛАДЬЕВ В.
20 INPUT "ЗАДАТЬ ТОМ С КАТАЛОГОМ(18F/18R/1CF/1CR)",D$
30 B$="S0SELECTDISKXXX":STR(B$,16,1)=HEX(85):INPUT #F1
40 STR(B$,13,3)=D$:ON ERROR E8$,E9%GOSUB450:LOAD B$%50,50,50
60 PRINT HEX(0312):PRINTUSING 470:PRINT :PRINTUSING 480
70 PRINTUSING 490:PRINTUSING 500:PRINTUSING 510
80 PRINTUSING 520:PRINT :PRINTUSING 530:GOSUB ' 80:INPUT J0
90 ON J0GOSUB340,210,120,190,100:GOTO 60
100 PRINT ,"ПРОГРАММА (SVEGAL) ЗАВЕРШЕНА":INPUT #F2
110 PRINT ,"TIME OF (SVEGAL)=";ROUND((F1-F2)/120000,1);" MIN":END
120 PRINT HEX(0312):INPUT "ЗАДАТЬ ПРИРАЩЕНИЕ КАТАЛОГА",Z
130 IF Z(=K4THEN170:PRINT ,"ОШИБКА ЗАКАЗА":STOP :GOTO 120
140 INPUT "ЗАДАТЬ ПРИРАЩЕНИЕ ТЕКУЩЕГО КОНЦА КАТАЛОГА",Z5
150 GOSUB ' 47(0):GOSUB ' 42(3):A=A+Z5
160 GOSUB ' 62(A):STR(A$(),3,2)=U$:GOSUB ' 67(0):GOTO 180
170 MOVE END T=K2+Z:K2=K2+Z:GOTO 140
180 RETURN ;И ИЗМЕНЕНИЕ РАЗМЕРОВ КАТАЛОГА ТОМА
190 INPUT "PRINT TO: 05(1) OR 0C(2)",H:ON HGOTO200:SELECT PRINT0C(80)
200 PRINT HEX(0312):GOSUB ' 0(2,K0,F$,K9,L9):GOSUB ' 3:RETURN
210 INPUT "НА СКОЛЬКО СЕКТОРОВ УВЕЛИЧИТЬ УК.КАТАЛОГА",K9
220 IF K4-K9)=0THEN250:IF K2-K1-K9)=0THEN260
230 PRINT ,"УВЕЛИЧИТЬ УК НЕВОЗМОЖНО ИЗ-ЗА ОТСУТСТВИЯ"
240 PRINT ,"СВОБОДНОГО МЕСТА НА ТОМЕ И В КАТАЛОГЕ":STOP :GOTO 60
250 E1=K2:E2=K0:E3=K2+K9:E9=E3:S=-1:L=1:GOTO 270
260 E1=K1:E2=K0:E3=K1+K9:E9=E3:S=-1:L=2
270 GOSUB ' 87(E1,E2,E3,S):ON LGOTO280:K1=K1+K9:K0=K0+K9:GOTO 290
280 K2=E9:K1=K1+K9:K0=K0+K9:K4=K4-K9:GOTO 290
290 INIT (HEX(00))A$():FOR K=E2TOE2+K9-1
300 GOSUB ' 67(K):NEXT K:GOSUB ' 47(0):GOSUB ' 62(K0)
310 STR(A$(),1,2)=U$:GOSUB ' 62(K1):STR(A$(),3,2)=U$
320 GOSUB ' 62(K2):STR(A$(),5,2)=U$:GOSUB ' 67(0)
330 GOSUB ' 0(3,K0,F$,K9,L9):GOSUB ' 1:RETURN
340 PRINT HEX(0312):INPUT "ЗАДАТЬ ИМЯ УДАЛЯЕМОГО ФАЙЛА",F$
350 GOSUB ' 0(1,K0,F$,K9,L9):IF W=1947THEN340
360 K9=R2-R1+1:L9=R1:K1=K1-K9:GOSUB ' 62(K1):GOSUB ' 47(0)
```

```
370 STR(A%(),3,2)=U%:GOSUB ' 67(0):S=1:E3=R1:E1=R2+1:E2=K1+K9:L9=R1
380 GOSUB ' 87(E1,E2,E3,S):GOSUB ' 0(4,K0,F%,K9,L9):GOSUB ' 1:RETURN
390 DEFFN ' 42(N):A%=STR(A%(),N,1):GOSUB 420:B=L:C=P
400 A%=STR(A%(),N+1,1):GOSUB 420:D=L:E=P
410 A=B*16^3+C*16^2+D*16^1+E:RETURN
420 M=VAL(A%):L=INT(M/16):P=ROUND((M/16-L)*16,0):RETURN
430 DEFFN ' 47(T):DATA LOAD BA T(T)A%():RETURN
440 DEFFN ' 67(T):DATA SAVE BA T(T)A%():RETURN
450 PRINT HEX(030712):PRINT ,"ATTN: СНЯТЬ ЗАЩИТУ С: ";U%:RETURN
460 PRINT HEX(030712):GOTO 60:% SYSTEM (BASIC 02)
470 %        ПРОГРАММА (SVEGAL) - ПОЛНАЯ МОДИФИКАЦИЯ КАТАЛОГА
480 % - ФИЗИЧЕСКОЕ УДАЛЕНИЕ ФАЙЛОВ С УПЛОТНЕНИЕМ ТЕЛА КАТАЛОГА   - 1
490 % - УВЕЛИЧЕНИЕ УКАЗАТЕЛЯ КАТАЛОГА ТОМА НА (G) СЕКТОРОВ       - 2
500 %,- УВЕЛИЧЕНИЕ ТЕКУЩЕГО И/ИЛИ ФИКСИРОВАННОГО КОНЦОВ КАТАЛОГА - 3
510 % - РАСПЕЧАТКА ПОЛНОГО СОСТОЯНИЯ КАТАЛОГА ЗАДАННОГО ДИСКА.   - 4
520 % - ЗАВЕРШЕНИЕ РАБОТЫ С  ПРОГРАММОЙ РЕОРГАНИЗАЦИИ - (SVEGAL) - 5
530 %                - УКАЖИТЕ НОМЕР ТРЕБУЕМОГО ДЕЙСТВИЯ (1-5)
540 DEFFN ' 80:T=0:GOSUB ' 47(T):N=1:GOSUB ' 42(N):K0=A:N=3
550 GOSUB ' 42(N):K1=A:N=5:GOSUB ' 42(N):K2=A
560 IF STR(D%,1,2)="1C"THEN570:K4=1000-K2:GOTO 580
570 K4=9791-K2:% ПОДСЧЕТ СВОБОДНОГО МЕСТА В ТОМЕ
580 RETURN :% THEORY OF HOMOGENEOUS STRUCTURES
590 DEFFN ' 87(E1,E2,E3,S):FOR K=E1TOE2STEP3:GOSUB ' 47(K)
600 GOSUB ' 67(E3):E3=E3+S:NEXT K:RETURN
610 DEFFN ' 62(U):DATA LOAD BA T(U-1,U%)S%:RETURN
620 DEFFN ' 0(P0,K0,F%,K9,L9):IF P0()2THEN630:GOSUB 930
630 G0=0:DIM M1%2:FOR K=0TOK0-1:GOSUB ' 47(K):W=0
640 FOR I=1TO239STEP16:M%=STR(A%(),I,16):M1%=STR(M%,1,2)
650 IF M1%()HEX(1080)ANDM1%()HEX(1180)ANDM1%()HEX(1100)ANDM1%()HEX(1000)THEN760
660 G0=G0+1:ON P0GOTO680,670,700,730
670 LIMITS TSTR(M%,9,8),R1,R2,R3,R4:GOSUB ' 45(M%,R1,R2,R3,R4):GOTO 760
680 IF STR(M%,9,8)()F%THEN760:INIT (HEX(00))M%:GOSUB ' 42(I+2)
690 R1=A:GOSUB ' 42(I+4):R2=A:STR(A%(),I,16)=M%:GOSUB ' 67(K):GOTO 820
700 GOSUB ' 42(I+2):R1=A:GOSUB ' 42(I+4):R2=A:R1=R1+K9
710 R2=R2+K9:GOSUB ' 62(R1):STR(A%(),I+2,2)=U%:GOSUB ' 62(R2)
720 STR(A%(),I+4,2)=U%:GOTO 760
730 GOSUB ' 42(I+2):R1=A:GOSUB ' 42(I+4):R2=A
740 IF R1(L9THEN760:R1=R1-K9:R2=R2-K9:GOSUB ' 62(R1)
750 STR(A%(),I+2,2)=U%:GOSUB ' 62(R2):STR(A%(),I+4,2)=U%:GOTO 760
760 NEXT I:ON P0GOTO770,770:GOSUB ' 67(K)
770 NEXT K:ON P0GOTO780,790:GOTO 820
780 PRINT ,"ФАЙЛА С ЗАДАННЫМ ИМЕНЕМ НЕТ В КАТАЛОГЕ":W=1947:GOTO 810
790 PRINT :PRINT ,"КОЛИЧЕСТВО  ФАЙЛОВ  В  КАТАЛОГЕ   ТОМА=";G0
800 PRINT ,"ЧИСЛО СВОБОДНЫХ МЕТОК ФАЙЛОВ В УК ТОМА=";K0*16-G0-1
810 GOSUB ' 3:STOP "ДЛЯ ПРОДОЛЖЕНИЯ РАБОТЫ НАЖАТЬ 'CONTINUE'"
820 SELECT PRINT05(80):RETURN
830 DEFFN ' 45(M%,R1,R2,R3,R4):CONVERT R1TOR1%,(####)
840 CONVERT R2TOR2%,(####):CONVERT R3TOR3%,(####)
850 ON R4GOTO860,870,880,890,900,910:R4%="ЗТП0":GOTO 920
860 R4%="П0":GOTO 920:%  ##########################################
870 R4%="0Д":GOTO 920:%  #   АЛАДЬЕВ В.Э. = СКБ  МПСМ  ЭССР    #
880 R4%="УП0":GOTO 920:% # ТАЛЛИН 200006,  МУСТАМЯЭ  ТЕЕ  55  #
890 R4%="У0Д":GOTO 920:% #       ТЕЛ. 527-665, 527-664          #
900 R4%="ТП0":GOTO 920:% # ТАЛЛИН 200035, ПАЛДИСКИ МНТ 171-26 #
910 R4%="ЗП0":GOTO 920:% ##########################################
920 PRINT STR(M%,9,8);" ";R1%;" ";R2%;" ";R3%;" ";R4%:RETURN
930 REM УКАЗАТЕЛЬ КАТАЛОГА:PRINT ,"СОСТОЯНИЕ КАТАЛОГА ТОМА"
940 PRINT :PRINT ,"РАЗМЕР УКАЗАТЕЛЯ КАТАЛОГА ТОМА     =";K0
950 PRINT ,"ТЕКУЩИЙ   КОНЕЦ  КАТАЛОГА ТОМА     =";K1
960 PRINT ,"ФИКСИРОВАННЫЙ КОНЕЦ КАТАЛОГА ТОМА  =";K2
970 PRINT ,"СВОБОДНОЕ МЕСТО ЗА КАТАЛОГОМ  ТОМА =";K4
980 PRINT :PRINT :RETURN :DEFFN ' 1:INIT (00)M%
990 FOR K=0TOK0-1:GOSUB ' 47(K):V9=1942
1000 FOR I=1TO222STEP16:IF STR(A%(),I,16)()M%THEN1030
1010 IF STR(A%(),I+16,16)=M%THEN1030
1020 STR(A%(),I,16)=STR(A%(),I+16,16):STR(A%(),I+16,16)=M%:V9=45
1030 NEXT I:IF V9()45THEN1040:V9=1967:GOTO 1000
1040 GOSUB ' 67(K):NEXT K:RETURN :DEFFN ' 3:PRINT
1050 PRINT ,"ЕСЛИ ДЛЯ НЕКОТОРЫХ ФАЙЛОВ КАТАЛОГА УКАЗАНЫ НУЛЕВЫЕ "
1060 PRINT ,"ЗНАЧЕНИЯ ХАРАКТЕРИСТИК,ТО СЛЕДУЕТ СДЕЛАТЬ ЛИСТИНГ  "
1070 PRINT ,"КАТАЛОГА ПО ДИРЕКТИВЕ  LIST DC(F/R/T)[/(ФАУ)] И ДЛЯ"
1080 PRINT ,"ПОСЛЕДУЮЩЕГО ОБРАЩЕНИЯ К ТАКИМ ФАЙЛАМ СЛЕДУЕТ ИС-  "
1090 PRINT ,"ПОЛЬЗОВАТЬ ОПЕРАТОРЫ LOAD DA (ТИП)[/(ФАУ),](АДРЕС) "
1100 PRINT ,"ИЛИ DATA LOAD DA (ТИП)[/(ФАУ),](АДРЕС)":RETURN :% END
```

В качестве еще одного примера рассмотрим задачу выгрузки программы на диск, минуя обработку ее в операционной среде БЕЙСИКа, когда требуется работать, скажем, с иными

диалектами языка БЕЙСИК. Суть задачи состоит в следующем. По основной программе запрашивается ввод очередной ПС. После получения текста строки она переписывается в символьный массив Оо¤( ) заданной пользователем размерности. Концом ввода текста программы служит комбинация FFF.

Сформированный массив Оо¤( ) затем переписывается на диск по указанному адресу, информируя пользователя о его местоположении и реально занимаемой длине. Листинг описанной программы имеет вид:

```
10 COM A%253,B%31,X%5:L=1:% ПРОЦЕДУРА ВЫГРУЗКИ ПРОГРАММ
20 INPUT "ЗАДАТЬ ДИСК,РАЗМ-ОСТЬ О0%(),ЕГО НАЧАЛО НА ДИСКЕ",D%,X%,A
30 B%="50DIM00%(XXXXX)1:SELECTDISKXXX":STR(B%,31,1)=HEX(85)
40 STR(B%,10,5)=X%:STR(B%,28,3)=D%:LOAD B%50,50,50
60 INIT (' ')A%:LINPUT "ВВОД СТРОКИ ПРОГРАММЫ/FFF",A%
70 IF STR(A%,1,3)="FFF"THEN90:STR(O0%(),L)=STR(A%,1,LEN(A%))
80 STR(O0%(),L+LEN(A%))=HEX(85):L=L+LEN(A%)+1:GOTO 60
90 DATA SAVE DA T(A,S)O0%():STOP "ВКЛЮЧИТЬ ПРИНТЕР - 'CONTINUE'"
100 PRINT /0C,"МАССИВ O0%() ПЕРЕПИСАН НА ДИСК: ";D%
110 PRINT /0C,"РЕАЛЬНАЯ ДЛИНА МАССИВА O0%() = ";LEN(O0%())
120 PRINT /0C,"ПРОГРАММНЫЙ МАССИВ ЗАПИСАН В СЕКТОРАХ: ";A;" - ";S-1:END
```

При работе с процедурами пользователю целесообразно оформлять их отдельно и помещать в библиотеку программ (каталог) для хранения. Рассмотрим пример организации работы с процедурами, создаваемыми на уровне помеченных подпрограмм.

Нумерация строк основной программы осуществляется с шагом 10. Процедура оформляется в виде помеченной подпрограммы, листинг которой имеет вид:

```
101 GOTO 109:DEFFN ' 42(X,Y):Z=SQR(X^2+Y^2)+INT(X/Y)
103 Z=ROUND(Z,2):PRINT /0C,"Z1=";Z
104 Z=ROUND((X#Y)/Z,2):PRINT ,"Z2=";Z
108 RETURN
109 REM КОНЕЦ ПРОЦЕДУРЫ
```

В каталог эта подпрограмма помещается под некоторым именем. Нумерация строк в процедуре осуществляется с шагом 1, что не нарушает целостности основной программы при загрузке процедуры в память ЭВМ. Чтобы процедура, в свою очередь, не нарушала естественного порядка выполнения основной программы, первая ее строка начинается с передачи управления на конец процедуры. Листинг основной программы имеет вид:

```
10 REM ИСПОЛЬЗОВАНИЕ БИБЛИОТЕЧНЫХ ПРОЦЕДУР:COM X,Y:X=12
20 Y=42.4767:LOAD DC F"DEFFN'42"101,101,40
40 GOSUB ' 42(X,Y):PRINT /0C,"X=";X;"Y=";Y
100 INPUT "РАБОТУ ПОВТОРИТЬ(1-ДА/2-НЕТ)",P:ON PGOTO120,130
120 X=X+1:Y=Y+0.5:GOTO 40
130 END :REM КОНЕЦ ФРАГМЕНТА ПРОГРАММЫ

Z1= 44.14
X= 12 Y= 42.4767
```

Если для оформления процедуры требуется более девяти строк, то можно использовать следующую ее организацию:

```
101 GOTO 109:DEFFN ' 42(X,Y):Z=SQR(X^2+Y^2)+INT(X/Y)
103 Z=ROUND(Z,2):PRINT /0C,"Z1=";Z
104 Z=ROUND((X#Y)/Z,2):PRINT /0C,"Z2=";Z
108 GOSUB ' 47:RETURN
109 REM КОНЕЦ ПРОЦЕДУРЫ:GOTO 119:DEFFN ' 47
112 PRINT "КОНЕЦ РАСЧЕТА ПО ПРОЦЕДУРЕ"
119 RETURN :% КОНЕЦ ВТОРОЙ ЧАСТИ ПРОЦЕДУРЫ
```

При рассмотрении помеченных и обычных подпрограмм уже отмечалось выше о возможности входа внутрь помеченной подпрограммы посредством оператора GOSUB. Следующий небольшой фрагмент иллюстрирует обращение с помощью оператора GOSUB в самое начало помеченной подпрограммы:

```
10 % ОПЕРАТОРЫ GOSUB ' И GOSUB :P=1:T=8:L=42
20 GOSUB ' 1(P,T,L):PRINT M;:GOSUB 40:PRINT M;:END
30 DEFFN ' 1(P,T,L):M=P+T+L:PRINT "GOSUB ' 1=";;:RETURN
40 DEFFN ' 1(P,T,L):M=P#T#L:PRINT "GOSUB ''1=";;:RETURN

GOSUB ' 1= 51 GOSUB ''1= 336
```

Различные способы обращения к помеченным подпрограммам позволяют очень гибко использовать это сильное языковое средство БЕЙСИКа (см. листинг программы "Интерпретатор").

Кратко опишем теперь несколько прикладных программных средств для ЭВМ "Искра-226".

Первоначально ППП "Контроль" был создан для организации контроля выполнения мероприятий Министерства промышленности строительных материалов ЭССР. Однако в процессе работы с ним выяснилось, что он при небольшой доработке с успехом может быть использован для более широкой организации автоматизированных систем контроля исполнительской дисциплины в различных организациях и ведомствах.

В организационном плане ППП "Контроль" является замкнутой системой и не требует использования каких-либо других сервисных средств, хотя последние разработаны в СКБ Министерства промышленности строительных материалов ЭССР [36]. ППП "Контроль" допускает несложную настройку на работу с другим смысловым содержимым реквизитов входных (выходных) форм документов. Наряду с функциями контроля ППП дает возможность пользователю документировать программные средства.

Все возрастающий объем проблемного ПО (приобретаемого и разрабатываемого) для любой организации, эксплуатирующей ВТ, ставит на повестку дня организацию службы каталогизирования ПО. Учитывая персональный характер использования ЭВМ "Искра-226", функцию ведения каталога ПО целесообразно возложить на саму ЭВМ. С этой целью была разработана самодокументируемая программа "Каталог", позволяющая организовать простую и вместе с тем достаточно удобную службу ведения каталога ПО ЭВМ "Искра-226" [31].

Программа "Каталог" допускает работу в следующих основных режимах: 1) создание каталога ПО; 2) корректировка каталога ПО; 3) вывод каталога ПО на АЦПУ или дисплей. В каталоге ПО под запись для конкретного программного средства отводится один сектор (256 байтов). Запись содержит 14 реквизитов, описывающих основные характеристики программного средства согласно табл. 11.

На АЦПУ выводится вся отредактированная запись о каждом программном средстве, включенном в каталог. На дисплее отображаются только реквизиты 13, 9, 2 и 3 каталогизированного программного средства, т.е. выводится сокращенная информация о каталогизированных программных средствах.

В ряде случаев у пользователя возникает потребность использования ЭВМ "Искра-226" на уровне инструкций ЭВМ (например, желание создать высокоэффективную специализированную систему обработки информации, получить в свое распоряжение максимум доступной ОП, работать с нестандартными внешними устройствами, повысить быстродействие ЭВМ и т. д.). Так, при работе в операционной среде БЕЙСИКа программа выполняется со средней скоростью около 400–500 операторов в секунду при средней производительности процессора около $5 \cdot 10^5$ инструкций ЭВМ в секунду, т. е. временной выигрыш при переходе на более низкий уровень программирования может быть существенным.

Таблица 11. Основные характеристики программного средства

Реквизит	Длина в байтах	Смысловое содержание
1	3	Номер программного средства
2	6	Имя НМД или НМЛ (идентификатор в системе хранения)
3	3	ФАУ (18F, 18R, 1CF, 1CR, 10)
4	9	Тип средства (сервис, программа, данные, ППП, система, пример и т. п.)
5	17	Версия операционной среды
6	4	Общее число занимаемых средством секторов
7	1	Признак защиты (Т – без защиты, Ст – защита)
8	3	Структура средства (ЕМ – единый модуль, ЕМД – ЕМ с данными, ММ – многомодульное средство, ММД – ММ с данными)
9	2	Номер версии программного средства (00-99)
10	8	Дата создания программного средства
11	20	Автор разработки программного средства
12	15	Организация–разработчик программного средства
13	8	Наименование программного средства
14	115	Краткое содержание и возможности программного средства

Для автоматизации разработки программ на языке инструкций ЭВМ создан транслятор с языка "Ассемблер-226", основой которого является набор мнемоник для инструкций ЭВМ "Искра-226". ППП "Ассемблер-226" предназначен для создания, хранения и сопровождения программ, разработанных на языке "Ассемблер" ЭВМ "Искра-226" [35]. ППП реализует следующие основные виды работ с программами на языке "Ассемблер-226":

ввод, трансляцию и редактирование программ;
архивные операции с программами;
документирование программ;
загрузку программ на выполнение.

Режим работы ППП определяется соответствующими управляющими операторами. Подробно работа с ППП "Ассемблер-226" описана в [35]. Пользователю этот пакет поставляется на НГМД и содержит в своем составе операционную среду с транслятором из языка "Ассемблер-226", а также руководство программиста, состоящее из трех отдельных файлов: AS MB-РУК, AS MB-1, AS MB-2.

Для вывода текста руководства программиста на широкую печать следует привести в рабочее состояние процессор, дисковод с НГМД и АЦПУ, входящие в состав ЭВМ "Искра-226". Затем на УВВ с ФАУ–18RR или 18F надо установить НГМД с ППП "Ассемблер-226" и на языке загрузчика ввести директиву

LOAD {R/F} #2 CR/LF.

После этого на дисплее появится сообщение

В.Шилко 01.09.83

Запуск ППП "Ассемблер-226" осуществляется директивой

RUN 65000 CR/LF,

и на дисплее индицируется сообщение

ASSEMBLER–226

после чего нужно ввести директиву файлового режима работы:

FILE CR/LF

На дисплее при этом появится сообщение

ЧИТАТЬ ФАЙЛ
ЗАПИСАТЬ ФАЙЛ
УДАЛИТЬ ФАЙЛ
ЗАМЕНИТЬ ФАЙЛ
РАСПЕЧАТАТЬ КАТАЛОГ
ИНИЦИИРОВАТЬ ТОМ
КОПИРОВАТЬ ТОМ

После этого надо нажать клавишу HALT/STEP, определяя режим чтения файла в ОП. Запрашивается имя тома, которое вводится в светящееся окно. Оставшаяся часть окна при этом

заполняется пробелами. ЭВМ производит поиск тома и запрашивает имя файла. Вводится имя файла AS МВ-РУК. Выход из файлового режима после считывания файла осуществляется нажатием клавиши CR/LF.

Файл, считанный в память ЭВМ, может быть распечатан по директиве LIST/OC. Нажатием клавиши RESET записанный файл стирается. Аналогичным образом считываются и выводятся на печать файлы ASMB-1 и ASMB-2, содержащие описание языка "Ассемблер-226" и примеры его применения.

После ознакомления с руководством программиста пользователь может приступить к непосредственной работе с ППП "Ассемблер-226".

В заключение рассмотрим ПО задачи "Метролог" [46], касающейся автоматизации измерения расхода жидкостей и газов стандартными сужающимися устройствами. Программная реализация данной задачи требует решения вопросов оптимального использования памяти ЭВМ и организации информационной базы (содержащей большое число значительных по объему и различных по структуре и режиму их использования справочников физических данных), развитой системы ее обслуживания, а также прямого доступа к этой базе и минимизации времени поиска необходимых справочных данных.

Так как ЭВМ "Искра-226" представляет пользователю только 64 Кбайта своей ОП, то ПО задачи "Метролог" было реализовано в виде многомодульной программной системы, состоящей из резидентного ядра и динамически загружаемых модулей. Ядро задачи постоянно находится в памяти ЭВМ и выполняет следующие основные функции:

инициирует решение задачи и управляет всем его ходом;

обрабатывает возникающие особые ситуации;

осуществляет прямой доступ к справочным данным;

вычисляет промежуточные данные методом линейной интерполяции;

осуществляет диалог с пользователем и контроль входных данных;

ведет службу времени на основе таймера ЭВМ;

обрабатывает данные отдельных модулей и выдает конечный результат.

По требованию пользователя ядро задачи динамически загружает в память ЭВМ (передает управление и отслеживает ход выполнения) один из восьми модулей, каждый из которых выполняет следующие функции.

Модуль МИНФРМ обеспечивает ведение информационной базы задачи, подготавливая и проверяя НМД и (или) НГМД, формируя справочную информацию томов (имя тома и каталог его справочников). Принципы организации ведения этой базы

можно успешно использовать при решении широкого круга задач на ЭВМ "Искра-226".

Модули МВЕТВЬк(к=1,5) реализуют алгоритмы расчета расхода жидкостей и газов методом стандартных сужающихся устройств (диафрагм) для пяти измеряемых сред: природного газа, сжатого воздуха, перегретого и насыщенного пара, а также воды. Каждый из этих модулей выполняет следующие основные функции:

вводит дополнительные данные для расчета (по мере надобности);

ведет расчет по основному алгоритму;

обрабатывает особые ситуации, возникающие в ходе расчета;

выводит нужное число унифицированных выходных форм с результатами расчета;

подсчитывает время, затраченное на выполнение расчета.

Алгоритмы расчета расхода жидкостей и газов реализованы в полном соответствии с руководящими материалами Госкомитета по стандартам СССР. После завершения работы модулей МВЕТВЬк управление снова получает ядро задачи "Метролог".

Модуль МВЕТВЬ6 обеспечивает вывод на печать нужного числа копий заготовок форм для исходных данных задачи по всем измеряемым средам. Все входные формы унифицированы и содержат наряду со всеми утверждающими и согласующими реквизитами наименования всех необходимых для данного расчета входных данных и их единиц, что позволяет значительно облегчить не только сбор и подготовку исходных данных, но и работу пользователя.

Модуль МВЕТВЬ7 функциональной нагрузки не несет и содержит лишь информацию для пользователя о требованиях к разработке модулей, работающих под управлением ядра задачи "Метролог". Он служит для реализации дальнейших расширений возможностей задачи.

Экспериментные оценки показали, что квалифицированный пользователь, пользуясь электронным микрокалькулятором, на расчет только одной измеряемой среды тратит не менее 3 ч. времени, тогда как ПО задачи "Метролог" позволяет сделать это за 3–5 мин., включая время ввода исходных данных с дисплея и вывод выходной формы с результатами на широкую печать. Следовательно, только временной выигрыш составляет около 40 раз, но это не единственное преимущество.

Использование ПО задачи "Метролог" дает возможность значительно повысить точность расчетов и снизить требования к квалификации пользователя, что особенно важно в условиях работы промышленного предприятия, оснащенного многими

измерительными устройствами для определения расхода его энергетических ресурсов. ЭВМ "Искра-226", позволяя подключать к ней аналоговые и цифровые датчики в качестве УВВ, дает возможность создавать АРМы для обеспечения не только метрологической службы предприятия, но и других служб.

7. МОДЕЛИРОВАНИЕ ДИСКРЕТНЫХ ПАРАЛЛЕЛЬНЫХ ДИНАМИЧЕСКИХ СИСТЕМ НА ЭВМ "ИСКРА-226"

7.1. Общие сведения

В данном разделе кратко описываются программные средства для моделирования дискретных ПДС, ориентированные на работу в операционной среде БЕЙСИКа. В последние годы значительно возрос интерес к различного типа дискретным ПДС и получен ряд важных результатов теоретического и прикладного характера [50–61]. В основе растущего интереса к таким ПДС лежит возможность использования их для моделирования широкого круга дискретных вычислительных устройств и процессов параллельного действия [50], а также использование их в качестве математических моделей перспективных вычислительных средств параллельного действия [58]. Однако теоретические исследования ПДДС встречают в ряде случаев существенные трудности. Поэтому все чаще имеют место попытки привлечь к их исследованию возможности ЭВМ. Однако до сих пор эти попытки носили эпизодический характер. Здесь предлагаются пути и средства широкого и систематического применения компьютерного моделирования для исследования дискретных ПДС. Используемые ниже понятия и определения можно найти в [52–61].

7.2. Использование локальных функций перехода

Изучение ряда классов дискретных ПДС как новой среды для математического моделирования [52–56] тесно связано с исследованием свойств так называемых локальных функций перехода, которые представляют собой k-значные логические функции (k-ЗЛФ). Среди различных подходов к исследованию таких функций особое место занимает подход с использованием алфавита A_k(k-простое), в котором каждая k-ЗЛФ может быть представлена полиномом по (mod k) максимальной степени n(k–1) над полем A_k и наоборот, где k–ЗЛФ есть любое отображение $R^{(n)}: A_k^n \to A_k$. В случае же составного k "почти все" k-ЗЛФ не могут быть представлены в алфавите A_k в такой полиномиальной форме [54].

В связи с этим возникает вопрос: можно ли определить алгебраическую систему (АС), которая допускала бы полино-

миальное представление k-ЗЛФ в алфавите A_k (k-составное) подобно случаю простого k.

Рассмотрим АС, в которой "почти все" k-ЗЛФ имеют полиномиальное представление для случая составного модуля k [57]. Такая АС определяется следующим образом. Пусть $A_k = (0,1,...,k-1)$ и на A_k определена обычная операция сложения по (mod k). Одновременно на A_k определяется бинарная операция $\otimes$ –умножения согласно следующей таблице умножения

$\otimes$	0	1	2	3	4	5	.....	(k–1)
0	0	0	0	0	0	0	.....	0
1	0	1	2	3	4	5	.....	(k–1)
2	0	2	3	4	5	6	.....	1
3	0	3	4	5	6	7	.....	1 2
4	0	4	5	6	7	8	.....	1 2 3
5	0	5	6	7	8	9	.....	1 2 3 4
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
(k–1)	0	(k–1)	1	2	3	.....		(k–2)

(1)

Легко убедиться в том, что операция $\otimes$ –умножения на множестве $A_k \setminus \{0\}$ образует конечную циклическую группу $A_k^{\otimes}$ степени (k–1). Имеет место следующее утверждение [54, 55].

Утверждение 4. Существует АС $\langle A_k; +; \otimes \rangle$, в которой "почти каждая" k-ЗЛФ в алфавите A_k (k-составное) может быть представлена в форме полинома $P_{\otimes}(n)$ (mod k), где:

1) (+) – операция сложения по (mod k);

2) ($\otimes$) – операция умножения, определяемая таблицей (1);

3) $P_{\otimes}(n) = \sum_{i=1}^{k^n-1} c_i \otimes x_1^{di_1} \otimes x_2^{di_2} \otimes ... \otimes x_n^{di_n} \pmod k$ –

полином, который не содержит двучленов вида $P_d \otimes X_j^d + B_d \otimes X_j^{k-d-1}$ $(0 \leqslant d_{i_j} \leqslant k-1; \sum_{j=1}^{n} d_{i_j} \geqslant 1; X_j, c_j \in A_k; P_d + B_d = k; P_d, B_d \geqslant 1; X_j^p = \underbrace{X_j \otimes X_j \otimes ... \otimes X_j}_{p}; j=\overline{1,n}; i=\overline{1, k^n-1}; d=\overline{1,[(k-2)/2]})$.

Это утверждение сыграло важную роль в исследовании динамических свойств дискретных ПДС в случае алфавита A_k (k-составное) и позволило получить целый ряд интересных результатов в теории d-мерных однородных структур (d–HS) [54, 55]. Более того, данное утверждение дает удовлетворительное аналитическое представление k-ЗЛФ в случае составного модуля k. Например, даже такая простая логическая функция, как

$$R_1^{(1)}(x) = \begin{cases} 0, \text{ если } x=0; \\ 2, \text{ если } x=1; \\ 1 - \text{в противном случае,} \end{cases}$$

в алфавите A_6 не может быть представлена полиномом по (mod 6), тогда как в АС $\langle A_6; +; \otimes \rangle$ ее представление имеет вид $R_1^{(1)}(x) = P_{\otimes}^1(1) = x^2 + x^3 \pmod 6$. Для сравнения эта же функция $R_1^{(1)}(x)$ в алфавите A_5 представляется как $R_1^{(1)}(x) = 4(x + x^2 + x^3) \pmod 5$. Логическая функция

$$R_2^{(1)}(x) = \begin{cases} 0, \text{ если } x=0 \text{ или } x=1; \\ 3, \text{ если } x=4; \\ x+1 - \text{в противном случае} \end{cases}$$

также не может быть представлена в полиномиальном виде при k=6, тогда как в АС $\langle A_6; +; \otimes \rangle$ и в алфавите A_5 ее представления имеют вид $R_2^{(1)}(X) = P_{\otimes}^2(1) = X + 2 \otimes X^2 + 3 \otimes X^3 \pmod 6$ и $R_2^{(1)}(X) = X^2 + 3(X + X^3 + X^4) \pmod 5$ соответственно.

Различия между представленной здесь АС $\langle A_k; +; \otimes \rangle$ и известной системой $\langle A_k; +; x \rangle$, в которой операции (+) и (x) являются обычными операциями сложения и умножения по (mod k), более существенны. Об этом свидетельствуют также следующие результаты [57]:

1. В АС $\langle A_k; +; \otimes \rangle$ ($k>3$) уравнение $X^p + Y^p = Z^p \pmod k$ для каждого целого $1 \leqslant p \leqslant k-2$ имеет ровно $M=](k-1)(k-3)/2[$ различных решений $X, Y, Z \in A_k \{0\}$, где $]T[$ – целая часть T, превышающая T, тогда как в случае АС с полиномиальной арифметикой по (mod k) число решений указанного уравнения для допустимых значений p не всегда достигает M.

2. Существенной оказывается зависимость свойств алгебраической системы $\langle A_k; +; \otimes \rangle$ от типа модуля k. Так, при простых k в системе имеет место соотношение

$$\sum_{i=1}^{k-1} x^i \pmod{k-1} = \begin{cases} k-[k/2]-1, \text{ если X четное;} \\ 0, \text{ если X нечетное,} \end{cases}$$

а при составных $k \sum_{i=1}^{k-1} x^i \pmod{k-1}=0$. Таким образом, тип модуля k играет существенную роль и в АС $\langle A_k; +; \otimes \rangle$ при полиномиальном представлении в ней k-ЗЛФ.

3. В АС $\langle A_k; +; x \rangle$ (k-простое) полиномиальное представление k-ЗЛФ однозначно, тогда как в системе $\langle A_k; +; \otimes \rangle$ данное условие в общем случае не выполняется. Так, для АС $\langle A_k; +; \otimes \rangle$ ($k \geqslant 6$) существует набор пар чисел $(p_1^1, p_1^2), (p_2^1, p_2^2), \ldots, (p_t^1, p_t^2)$ таких, что выполняются следующие соотношения:

$$X^{p_1^1} + X^{p_1^2} = X^{p_2^1} + X^{p_2^2} = \ldots = X^{p_t^1} + X^{p_t^2} = R_0^{(1)}(X) \pmod k,$$

где

$$R_0^{(1)}(x) = \begin{cases} 0, \text{ если } x=0; \\ 2, \text{ если } x=1; \\ 1 - \text{в противном случае;} \end{cases}$$

$1 = p_1^1 < p_i^1, p_i^2 < p_1^2 = k-2 \quad (i=\overline{2,t})$.

Например, $x + x^4 = x^2 + x^3 = R_0^{(1)}(x) \pmod 6$; $x + x^9 = x^3 + x^7 = R_0^{(1)}(x) \pmod{11}$; $x + x^{18} = x^2 + x^{17} = x^3 + x^{16} = \ldots = x^9 + x^{10} = R_0^{(1)}(x) \pmod{20}$.

4. В АС $\langle A_k; +; \otimes \rangle$ в отличие от системы $\langle A_k; +; x \rangle$ не соблюдается закон дистрибутивности: $X \otimes (Y+Z) \neq X \otimes Y + X \otimes Z \pmod k$. Например, в АС $\langle A_6; +; \otimes \rangle$ $4 \otimes (5+3) \neq 4 \otimes 5 + 4 \otimes 3 \pmod 6$. Следовательно, полиномы $R^{(2)}(X\ Y) = X \otimes (X+Y) \pmod k$ и $R_2^{(2)}(X,Y) = X^2 + X \otimes Y \pmod k$, например, представляют в АС $\langle A_k; +; \otimes \rangle$ две различные логические функции двух переменных.

Данное свойство можно использовать для представления в полиномиальной форме k-ЗЛФ. Таким образом, в АС $\langle A_k; +; \otimes \rangle$ наряду с операциями (+), ($\otimes$) может использоваться и операция взятия скобок для представления полиномами k-ЗЛФ. Однако каждый полином в системе $\langle A_k; +; \otimes \rangle$, содержащий скобки, может быть записан в той же АС в обычной бесскобочной форме. Например, в АС $\langle A_4; +; \otimes \rangle$ $x \otimes (2 \otimes x + 2 \otimes x^3) = 3 \otimes x + 2 \otimes x^2 + 3 \otimes x^3 \pmod 4$.

Следовательно, операция взятия скобок не добавляет ничего нового для возможности полиномиального представления k-ЗЛФ в АС $\langle A_k; +; \otimes \rangle$. Однако недистрибутивность АС $\langle A_k; +; \otimes \rangle$ в значительной мере усложняет проведение основных операций в полиномиальной арифметике, опеределяемой данной системой. Этот и ряд других моментов сделали актуальным разработку специального программного пакета для автоматизации работы исследователя с АС $\langle A_k; +; \otimes \rangle$.

Разработанный для ЭВМ "Искра-226" программный пакет позволяет не только автоматизировать процесс представления произвольных k-ЗЛФ в полиномиальной форме $P_{\oplus}(n) \pmod k$, но и выполнять необходимые базовые операции в полиномиальной арифметике, определяемой АС $\langle A_k; +; \otimes \rangle$. Пакет разработан в операционной среде БЕЙСИКа и легко переносим на ЭВМ "Искра-1030" и все совместимые с IBM PC отечественные персональные и мини-ЭВМ. При разработке пакета использован целый ряд интересных возможностей языка БЕЙСИКа ИСКРА-226, позволивших существенно упростить его структуру и функционирование. Выше некоторые из этих возможностей были описаны. Здесь мы отметим только одну из использованных возможностей языка, которая состоит в следующем.

В операционной среде БЕЙСИКа ЭВМ ИСКРА-226 существует возможность рекурсивного обращения к массивам (матрицам и векторам), когда в качестве индексов массива можно использовать различной глубины конструкции типа

$$M(A\,1(A\,2(\ldots(A_k(B, C\,1(C\,2(\ldots(C_k(K, P))\ldots).$$

Глубина рекурсии ограничивается лишь размером доступной ОП. Нижеследующий фрагмент пакета иллюстрирует пример

рекурсивного обращения к двухмерному массиву M (·), что позволяет весьма эффективно организовать не только операцию ⊗–умножения и вычисления степеней в АС $\langle A_k; +; \otimes \rangle$, но и существенно упростить реализацию базовых операций с полиномами вида $P_{\otimes}$ (n) (mod k) в полиномиальной арифметике, определяемой АС:

```
0 DIM M(40,40),N(40,40):CLEAR V:INPUT  задать (k)',K

1 MAT REDIM M(K-1,K-1),N(K-1,K-1):FOR J=1 TO K-1

2 M(1,J)=J:NEXT J:FOR I=2 TO K-1:FOR J=1 TO K-1

3 M(I,J)=I+J-1:NEXT J:FOR J=K-I+1 TO K-1:U=U+1

4 M(I,J)=U:NEXT J:U=0:NEXT I:MAT PRINT M:FOR I=1 TO K-1

5 N(I,1)=M(I,1):NEXT I:FOR I=1 TO K-1:FOR J=2 TO K-1

6 N(I,J)=M(N(I,1),N(I,J-1)):NEXT J:NEXT I:MAT PRINT N

7 INPUT  работу повторить(1-Д/2-Н)',U:ON U GOTO 0:END
```

Возможность рекурсивного обращения к массивам позволяет в операционной среде БЕЙСИКа довольно эффективно организовать в программах пользователя работу с такими важными объектами, как:

– структуры списочного типа,
– массивы размерности, большей двух, а также
– таблицы со сложными функциональными зависимостями.

Все это существенно расширяет выразительные средства языка БЕЙСИК.

7.3. Использование интерактивной программной системы

Для многоцелевого исследования теоретических и прикладных аспектов d–HS как одного из классов дискретных ПДС разработана интерактивная программная система (ИПС), которая представляет собой интегрированный интерактивный пакет для ЭВМ "Искра-226" на алгоритмическом языке БЕЙСИК. Лингвистическая реализация пакета делает переносимым его на многие ПК и мини-ЭВМ. Он предназначен для широкого круга пользователей, занимающихся теорией ОС и ее многочисленными приложениями.

Существенной частью пакета является ИПС, которая позволяет пользователю легко контролировать и управлять ходом вычислений так, что он может посвятить большинство своей интеллектуальной энергии собственно исследованию

интересующих его проблем: принятию решений, наблюдению исследуемых феноменов, проверке и выдвижению гипотез и т. д. ИПС состоит из структур данных, программ и диалоговых языков, специально предназначенных для удобства работы пользователя.

Первая версия пакета ИПС в основном завершена и предназначена для экспериментальных исследований d–HS и их приложений в ВТ, комбинаторике, математической биологии и биологии развития. Пакет открыт для расширения как круга исследуемых проблем, так и для пополнения его новыми типами d–HS. Использование пакета показало, что он и его дальнейшие модификации могут стать сильным инструментом изучения ряда классов дискретных ПДС.

Рассмотрим более детальнее архитектуру и возможности пакета ИПС, который содержит шесть основных компонентов: ядро; программы для исследования классических d–HS(PCd–HS), d–HS, с рефрактерностью (PRd–HS), недетерминированных d–HS(PNDd–HS), стохастических d–HS(PSd–HS) и полигенных d–HS(PPd–HS). Таким образом, пакет охватывает практически все основные типы d–HS. Ядро и пять вышеперечисленных программ образуют соответственно первый и второй уровни пакета. Третий его уровень составляют программы HSX(J) (J= 1, X; X ∈ (A, B, C, D, E)), предназначенные для решения тех или иных исследовательских проблем для соответствующих типов d–HS.

Ядро является резидентной программой, которая реализует все прямо необходимые и часто используемые функции пакета. Так, ядро содержит часто используемые модули программ второго уровня. Каждая компонента ядра является высоконадежным и оптимально закодированным модулем. Например, модуль реализации локальной функции перехода в d–HS основывается на следующем оптимальном алгоритме. На основе диалога с пользователем, формульного вычисления или считывания с диска формируется вектор W[1:a^n], содержащий новые состояния единичных автоматов d–HS согласно локальной функции перехода. Тогда состояние c(i, t) единичного автомата d–HS в момент t вычисляется по формуле

$$c(i,t) = W\left[1 + \sum_{j=1}^{n} c(j,t-1)\, a^{n-j}\right],$$

где c(j, t–1) – состояния всех соседей i-автомата ($j=\overline{1,n}$) в момент T–1. Ядро включает в себя модули сбора статистики о работе пакета и проводимых с ним процедурах. Так, по данным статистики время моделирования k-го шага классических d–HS составляет:

для 1–HS $T(k) = A(x)(k-1) + S(x,L);$

для d–HS $T(k) = B(d,x,L)\, k^d \quad (d > 1),$

где A(x) ⟩ 0.01(3x + 2)x, S(x,L) и B (d,x,L) – возрастающие по x, L и d функции; x, L – размеры шаблона соседства и начальной конфигурации структуры соответственно. Следовательно, в зависимости от быстродействия ЭВМ класс проблем, охватываемых пакетом, имеет ограничения как по размеру исследуемой активной области, так и по глубине изучаемой динамики d–HS.

Ядро управляет вычислительными ресурсами, порядком выполнения программ второго уровня (PCd–HS, PRd–HS, PPd–HS, PSd–HS, PNDd–HS) и осуществляет настройку пакета согласно требованиям пользователя. Ядро обеспечено хорошо развитым набором команд, которые образуют интерактивный управляющий язык – первый уровень лингвистического окружения пакета, позволяющий достаточно легко генерировать пакет по запросу пользователя и расширять его новыми программными возможностями.

Программы второго уровня пакета служат для компьютерного исследования основных типов d–HS. Каждая из вышеперечисленных пяти программ имеет управляющую часть, содержащую наиболее часто используемые модули. Эта часть реализует все часто используемые функции, связанные с исследованием соответствующего типа d–HS. Другие модули программы загружаются динамически. Исследуя d–HS вышеперечисленных пяти типов, пользователь работает с соответствующими программами второго уровня посредством специального интерактивного управляющего языка (СИУЯ). СИУЯ в значительной мере отражает как специфические черты исследуемых d–HS, так и используемую ими терминологию. В рамках СИУЯ может быть расширено само множество исследуемых по данной d–HS проблем. СИУЯ образует второй уровень лингвинистического окружения пакета.

Наконец, программы HSX(j) третьего уровня пакета служат для исследования конкретных проблем d–HS того или иного типа. Каждая программа третьего уровня снабжена, как правило, простым проблемным интерактивным управляющим языком (ППУЯ), который учитывает в деталях специфические черты исследуемой проблемы. Множество ППУЯ составляет третий уровень лингвинистического окружения пакета.

Иерархическая модульная структура программного и лингвинистического окружений пакета ИПС позволяет достаточно легко как использовать его возможности малоподготовленным пользователем, так и расширять его программными средствами для исследования новых перспективных проблем в теории ОС. Наконец, разработка и эксплуатация пакета ИПС позволили в значительной мере уяснить важные вопросы, связанные с моделированием дискретных ПДС на ЭВМ последовательного действия, и рассмотреть ряд проблем разработки параллельного ПО.

ЗАКЛЮЧЕНИЕ

Помещенные в настоящей книге материалы по архитектуре и ПО относятся в первую очередь к ЭВМ "Искра-226", которая с успехом может использоваться для решения не только отдельных локальных задач пользователя, но и других задач, ранее считавшихся привилегией больших ЭВМ универсального и специального назначения. В связи с дальнейшим развитием ВТ можно выделить ряд направлений, где без этих материалов также трудно обойтись.

Во-первых, серия ЭВМ "Искра-226" пополняется новыми модификациями и КПТС, к которым относятся ЭВМ "Искра-226 М", "Искра-226СОТ", АРМ технолога-программиста, где (как и в ЭВМ "Искра-1030" – последующей модификации ЭВМ "Искра-226") в основу ПО положен алгоритмический язык БЕЙСИК.

Во-вторых, в выпускаемых и планируемых к выпуску отечественных ПК ("ЕС-1840", "ЕС-1841", "ЕС-1800" и др.) в качестве основного языка программирования выбран тот или иной диалект БЕЙСИКа. Программа "Интерпретатор", описанная в данной книге, позволяет пользователю сравнительно несложно организовать перевод программ из одного диалекта языка БЕЙСИК на другой.

В-третьих, ПО ряда приобретаемых различными ведомствами и организациями страны зарубежных ПК разработано также на диалектах языка БЕЙСИК, весьма близких к БЕЙСИКу версии ЭВМ "WANG-2200MUP", который послужил аналогом описанной в настоящей книге версии языка БЕЙСИК ЭВМ "Искра-226". Это в первую очередь относится к ПК "Роботрон-1715", который в большом количестве начал поступать в нашу страну из ГДР. Ввиду отсутствия широкодоступных материалов по этому ПК данная книга в определенной мере восполнит этот пробел.

Наконец, язык БЕЙСИК, являясь одним из основных и наиболее популярных языков программирования мини-, микро-ЭВМ и ПК, составляет значительную часть курсов по программированию и информатике. И в этом плане изложенный на доступном широкому кругу читателей уровне язык БЕЙСИК представит интерес для слушателей курсов по программированию и информатике, студентов, учащихся и всех спе-

циалистов, использующих в своей деятельности персональные ЭВМ.

Материалы, помещенные в книге, касаются исключительно стандартно поставляемой ЭВМ "Искра-226" разного исполнения. Опыт работы с этой ЭВМ показал, что относительно несложные доработки позволяют существенно расширить ее возможности. Так, при совместном использовании устройств прямого доступа (НМД, НГМД) можно организовывать простые и вместе с тем эффективные вычислительные локальные сети и комплексы обработки информации. В этом случае не только возрастают суммарные вычислительные возможности отдельно работающих ЭВМ, но и появляется реальная возможность использования нетрадиционного способа обработки информации – параллельного, который способствует существенному повышению реальной скорости обработки информации различного назначения.

В качестве СПО таких комплексов можно было бы использовать параллельные системы обработки информации (ПСОИ), подобные тем, которые были созданы для многомашинных комплексов на основе моделей ЕС ЭВМ [9,12]. Снабжение ЭВМ "Искра-226" адаптерами типа "канал–канал" позволило бы существенно повысить эффективность подобных многомашинных ПСОИ на основе комплексов ПК.

Реализация указанных технических и программных доработок ЭВМ "Искра-226" и ее последующих модификаций позволит существенно расширить не только вычислительные возможности этого класса ЭВМ, но и сферу ее многочисленных применений в народном хозяйстве.

Наконец, кратко изложим современные тенденции развития и применения ПЭВМ на основе результатов проведенной в августе 1987 г. в США VI международной конференции по математическому моделированию. По представительности (свыше 5000 участников из 78 стран) и охвату рассматриваемой тематики (практически все современные разделы теоретических и прикладных наук) конференция по своему статусу соответствовала конгрессу. Все доклады, сделанные на конференции, опубликованы в [59]. Естественно, здесь невозможно дать полную характеристику столь важного научного форума. Отметим лишь материалы, характеризующие своего рода новые тенденции, связанные в первую очередь с применением ПК.

Наша страна на конференции была представлена двумя докладами. В нашем докладе [50] были рассмотрены вопросы теории и применения ОС в качестве математической модели для перспективных поколений высокопроизводительных параллельных вычислительных систем. В Массачусетском технологическом институте (США) именно на основе такой

модели была впервые реализована высокопараллельная вычислительная система CAM-6, на которой для ряда классов задач получено быстродействие около 45 млрд. оп/с.

Целый ряд докладов и сообщений был посвящен исследованию работы мозга и использованию его принципов для создания новых поколений высокоинтеллектуальных вычислительных систем и роботов, а также нетрадиционным применениям ВТ.

Ориентированные первоначально на решение относительно несложных задач в последние годы ПК все интенсивнее вступают в конкуренцию с мини-ЭВМ и ЭВМ более старших классов. Такая тенденция была отмечена именно на данной конференции. Принимая во внимание быстрый рост производства и областей применения ПК, организаторы конференции впервые включили в программу ее работы специальную секцию по ПК. Из результатов работы данной секции можно сформулировать ряд тенденций дальнейшего развития и применения ПК.

Прежде всего, массовый ПК ближайших лет представляется как 64-разрядная ЭВМ, имеющая от 1 до 16 Мбайт памяти, тем или иным способом распределенной между оперативной и внешней памятью. Быстродействие такого ПК будет не менее 2-3 млн. оп./с и он почти всегда будет соединен линиями связи с подобными ему или другими ЭВМ. Доступность таких линий связи будет столь же важной, как и доступность самих ПК и их ПО. Создание сетей обработки информации, включающих в себя в качестве элементов того или иного назначения ПК с вышеуказанными характеристиками, позволит массовому пользователю использовать все программные и информационные ресурсы такой информационно-вычислительной сети. Проведенный анализ показывает, что поставленным требованиям с некоторыми допущениями на сегодняшний день из массовых отечественных ЭВМ отвечают "Искра-226", "Искра-1030", ЕС-1841 и некоторые другие. Уже с учетом таких возможностей эти ЭВМ могут использоваться для создания удаленных АРМов, что существенно расширит возможности применения данных вычислительных средств.

Примером практического использования ЭВМ "ИСКРА-226" в вычислительных сетях может служить разработанная в Институте кибернетики АН УССР многоуровневая терминальная система, объединяющая мини-ЭВМ "СМ-4" и ПК "ИСКРА-226" с сетью ЕС ЭВМ на основе системы протоколов, реализованных в однородной вычислительной сети коммутации пакетов.

Вторая тенденция состоит в том, что рост программных и технических возможностей ПК позволяет использовать

их в тех сферах и для тех задач, которые ранее являлись привилегией ЭВМ более старших классов. Примерами такого рода могут служить задачи генетики, экономики, математического моделирования, прикладной и чистой математики, образования, математической биологии и биологии развития, социологии, медицины, оптимизации, техники, анализа управляющих систем, обороны и т. д. При определенных допущениях подобные задачи могут успешно решаться и на отечественных ЭВМ ("Искра-226", "Искра-1030", "ЕС-1841").

Третьей тенденцией является использование ПК в качестве непосредственной основы для моделирования. В этом случае некоторая модель непосредственно погружается в систему взаимосвязанных ПК, когда функционирование отдельного или ряда ПК системы моделирует работу того или иного блока, элемента и тому подобное исследуемого объекта. Реализация данной тенденции требует от ПК повышенной надежности, развитых средств сопряжения и специального ПО.

Наконец, четвертая тенденция состоит в смещении акцентов на разработку программных средств. В последние годы доля разработанных и разрабатываемых программных средств для ПК растет чрезвычайно быстрыми темпами. Так, общее число пакетов программ для ПК за рубежом уже превысило 20 тыс. При этом не учитывается пока все то ПО, которое на сегодняшний день создано большим количеством пользователей индивидуальных ПК в домашних условиях. А ведь среди них имеются весьма оригинальные разработки теоретического и прикладного характера.

В условиях существования региональных или национальных информационно-вычислительных сетей появляется возможность упорядочения и доступности программных средств индивидуального пользователя такой сети. Однако уже сейчас персональный характер использования ВТ определяет смещение акцентов в ее применении. Примером тому может служить разработка целого ряда весьма важных программных средств, ориентированных в первую очередь на различного типа ПК. Так, созданная под руководством Д. Кнута из Станфордского университета (США) программная система TEX для ПК явилась причиной того, что она принята в качестве эталонного программного средства подготовки материалов к публикации для вновь созданного международного журнала " Applied Mathematical letters ", срок публикаций в котором не будет превышать благодаря ее использованию трех месяцев со дня получения материалов редакцией.

Таким образом, класс ПК, как никакие другие предыдущие ЭВМ, не только настоятельно требует изменения технологий в целом ряде областей человеческой деятельности, но и создает

для этого основные предпосылки. И этой проблеме следует уделить самое пристальное внимание, учитывая ближайшие перспективы развития ВТ и в нашей стране, когда акцент по массовому использованию ВТ все больше будет смещаться именно на класс ПК. Появление за рубежом новейших 32-битных процессоров, функциональные возможности которых превосходят возможности процессоров крупных компьютеров недавнего прошлого, с особой остротой поднимает вопрос переосмысления в деле использования современной ВТ.

ЛИТЕРАТУРА

1. Аладьев В. З., Осипов О. Б. Введение в операционную систему ЕС ЭВМ, Таллин, 1975 г., Валгус, 180 стр.
2. Аладьев В. З., Осипов О. Б. Введение в архитектуру модулей ЕС ЭВМ, Таллин, 1976 г., Валгус, 310 стр.
3. Математическое обеспечение ЕС ЭВМ и АСУ (под редакцией В. З. Аладьева), Таллин, 1978 г., Валгус, 190 стр.
4. Система управления базами данных на основе операционной системы МИНИОС и СУБД ОКА (под редакцией В. З. Аладьева), Таллин, 1980 г., Валгус, 185 стр.
5. Параллельная обработка информации и параллельные алгоритмы (под редакцией В. З. Аладьева). Таллин, 1981 г., Валгус, 290 стр.
6. Аладьев В. З. Архитектура и программное обеспечение СМ ЭВМ. Таллин, 1983 г., ВЦ ЭРК Госбанка СССР, 87 стр.
7. Параллельные системы обработки информации (под редакцией В. З. Аладьева). Таллин, 1983 г., Валгус, 340 стр.
8. Процессор интерпретирующий диалоговый ЭВМ "Искра-226". Техническое описание и инструкция по эксплуатации 3.050.206.ТО. Курск, 1984 г., издательство Курского облисполкома, 54 стр.
9. ЭВМ "Искра-226". Описание ВМО "БЕЙСИК" – интерпретатора 1.320.136 ТОI. Курск, 1984 г., издательство Курского облисполкома, 49 стр.
10. ЭВМ "Искра-226". Справочное руководство по языку "БЕЙСИК". 1.320.136.Д14. Курск, 1984 г., издательство Курского облисполкома, 65 стр.
11. ЭВМ "Искра-226". Инструкция по программированию. Базовый объем. 1.320.136.Д14-1. Курск, 1984 г., издательство Курского облисполкома, 97 стр.
12. ЭВМ "Искра-226". Диски. Инструкция по программированию 1.320.136 Д14-2. Курск, 1984 г., издательство Курского облисполкома, 49 стр.
13. ЭВМ "Искра-226". Матрицы. Инструкция по программированию. 1.320.136 Д14-3. Курск, 1984 г., издательство Курского облисполкома, 29 стр.
14. ЭВМ "Искра-226". Сортировка. Инструкция по программированию. 1.320.136 Д14-4. Курск, 1984 г., издательство Курского облисполкома, 16 стр.
15. ЭВМ "Искра-226". Преобразование данных. Инструкция по программированию. 1.320.136 Д14-5. Курск, 1984 г., издательство Курского облисполкома, 37 стр.
16. ЭВМ "Искра-226". Графика. Инструкция по программированию. 1.320.136 Д14-6. Курск, 1984 г., издательство Курского облисполкома, 29 стр.
17. ЭВМ "Искра-226". Входной язык загрузчика. Инструкция по программированию. 1.320.136 Д14-7. Курск, 1984 г., издательство Курского облисполкома, 24 стр.
18. ЭВМ "Искра-226". Уточнение реализации языка "БЕЙСИК". Версия 01. Курск, 1984 г., издательство Курского облисполкома, 42 стр.
19. ЭВМ "Искра-226". Комплект контрольных тест-программ.

Библиотека программ. 1.320.136. Д15. Курск, 1984 г., издательство Курского облисполкома, 53 стр.

20. ЭВМ "Искра-226". Демонстрационные программы. Библиотека программ 1.320.136.Д15-1. Курск, 1984 г., издательство Курского облисполкома, 62 стр.

21. ЭВМ "Искра-226". Математика. Библиотека программ 1.320. 136.Д15-2. Курск, 1984 г., издательство Курского облисполкома, 96 стр.

22. ЭВМ "Искра-226". Статистика. Библиотека программ 1.320. 136.Д-3. Курск, 1984 г., издательство Курского облисполкома, 39 стр.

23. ЭВМ "Искра-226". Телекоммуникальный интерфейс. Библиотека программ 1.320.136 Д-4. Курск, 1984 г., издательство Курского облисполкома, 57 стр.

24. Система ввода, накопления, отображения и печати информации. Рига, 1984 г., НИИП, 128 стр.

25. Система подготовки и обработки текстов. Рига, 1984 г., НИИП, 48 стр.

26. Рыбаков В. И., Бессонов В. Ф. Система НИВА-1. Москва, 1983 г., АПК Госплана СССР, 120 стр.

27. Аладьев В. З. К теории однородных структур. Таллин, 1972 г., АН ЭССР, 259 стр.

28. Аладьев В. З., Шиленко В. Ф., Мартыненко Я. Г. Теоретические и прикладные аспекты однородных структур, в: Цифровые и аналоговые вычислительные системы. Таллин, 1987 г., Валгус, стр. 124–144.

29. Автоматизированная система управления ЭРК Госбанка СССР. Таллин, 1983 г., ВЦ ЭРК Госбанка СССР, 88 стр.

30. Аладьев В. З. Задача "Контроль". Описание и инструкция по эксплуатации. Таллин, 1986 г., СКБ МПСМ ЭССР, 30 стр.

31. Аладьев В. З. Задача "Каталог". Описание и инструкция по эксплуатации. Таллин, 1986 г., СКБ МПСМ ЭССР, 24 стр.

32. Газогареев М. И. Комплекс программ управления данными в информационных системах "Пудис". Руководство пользователя. Ленинград, 1983 г., ЛЭИС, 64 стр.

33. Ларионова Е. В. и др. Диалоговая система обработки таблиц. Методическое руководство. Москва, 1984 г., Госплан СССР, 108 стр.

34. Дудников Е. Е. Развитие персональных компьютеров. Москва, 1986 г. Приборы и системы управления, № 1, ЦНТЭИ приборостроения, стр. 56–86.

35. Шилко В. Н. Пакет программ АССЕМБЛЕР-226. Руководство программиста. Ленинград, 1985 г., Министерство приборостроения, средств автоматизации и систем управления СССР, 72 стр.

36. Аладьев В. З. Описание программных средств для персонального компьютера "Искра-226". Таллин, 1986 г., СКБ МПСМ ЭССР, 64 стр.

37. Система "ВАНГ-2200В". Руководство для пользователей. Рига, 1974 г., "ЗИНАТНЕ", 124 стр.

38. Кетков Ю. Л. Программирование на языке БЕЙСИК. Москва, 1978 г., Статистика, 186 стр.

39. Уорт Т. Программирование на языке БЕЙСИК. Москва, 1981 г., Машиностроение, 220 стр.

40. Программирование на языке БЕЙСИК-плюс для СМ-4. Москва, 1982 г., Финансы и статистика, 210 стр.

41. Аносов В. И. и др. Диалоговое программирование на языке БЕЙСИК. Методические указания по использованию мини-ЭВМ "Искра-226" в организациях Минтяжстроя СССР. Москва, 1983 г., Министерство строительства предприятий тяжелой индустрии СССР, 228 стр.

42. Математическое обеспечение ЕС ЭВМ и АСУ (под редакцией В. З. Аладьева). Таллин, 1978 г., Валгус, 190 стр.
43. Аладьев В. З. Сервисные программные средства для персонального компьютера "Искра-226". Таллин, 1986 г., СКБ МПСМ ЭССР, 56 стр.
44. Макглин Д. Микропроцессоры. Технология, архитектура и применение. Москва, 1979 г., Энергия, 328 стр.
45. ПТК ведения без данных ВЕЛОБАД для ЭВМ "Искра-226". Ленинград, 1984 г., НТПО Ленсистемотехника, 112 стр.
46. Аладьев В. З. Задача "Метролог". Описание и инструкция по эксплуатации. Таллин, 1986 г., СКБ МПСМ ЭССР, 76 стр.
47. Аладьев В. З. Решение ряда проблем теории однородных структур. Таллин, 1985 г., П/о "Силикаат", 84 стр.
48. Языки программирования БЕЙСИК и ПАСКАЛЬ. Москва, 1984 г., Алгоритмы и программы № 10, 144 стр.
49. Языки программирования и компилятор как средство взаимодействия с ЭВМ. Москва, 1984 г., Алгоритмы и программы № 10, 144 стр.
50. Aladyev V. Z. Homogeneous structures in mathematical modelling. Proc. the 6-th Intern. Conf. on Mathem. Modelling (August 1987), Saint-Louis, USA, 326–346 p.
51. Aladyev V. Z. Recent results in the theory of homogeneous structures. Proc. of the 4-th Conf. on Computer sciences. MTA. Budapest 1986, 221–238 p.
52. Aladyev V. Z. Mathematical Theory of Homogeneous Structures and Their Applications. Tallinn, 1980, Valgus, 268 p.
53. Aladyev V. Z. New results in the theory of homogeneous structures. Beitrage zur Theorie der Polyautomaten. Informatik-Skripten 8, Braunschweig 1984, 3–15 p.
54. Aladyev V. Z. Recent results on the theory of homogeneous structures, in: Trends, Tecniques and Problems in Theoretical Computer Science. Lecture Notes in Comp. Sci., Springer-Verlag 1987, 110–128 p.
55. Aladyev V. Z. Recent results in the homogeneous structures, in: Parallel Processing by Cellular Automata and Arrays, North-Holland 1986, 224–272 p.
56. Toffoli T., Margolus N. Cellular Automata Machines. A New Environment for Modeling. The MIT Press. Cambridge 1987.
57. Aladyev V. Z. An Algebraical System for Polynomial Representation of K–Valued Logical Functions. Appl. Mathem. Letters. vol. 1, no. 4 (1988), 124–128 p.
58. Aladyev V. Z. Survey of Some Theoretical Results and Applicability Aspects in Parallel Computation Modelling. J. New Generation Comp. Sys., vol. 1, no 4 (1988), 60–85 p.
59. Book of Abstracts. The 6-th Intern. Conf. on Mathem. Modelling (August 1987), Saint-Louis, USA.
60. Aladyev V. Z., Shilenko V. F. Computer Modelling of Homogeneous Structures. Informatik-Skripten 12, Brauschweig 1988.
61. Aladyev V. Z. A Solution of Steninhays's Combinatorical Problem. Appl. Mathem. Letters, vol. 1., no. 1 (1988), 11–12 p.
62. Lin Yn-Cheng and Gibson G., Microcomputer Systems: The 8086/8088 Family. Prentice-Hall, Englewood Cliffs, N. I. 1984.

ОГЛАВЛЕНИЕ

Предисловие 3
Список сокращений 5

1. АРХИТЕКТУРА СЕМЕЙСТВА ЭВМ „ИСКРА-226" 7

1.1. Назначение и основные технические данные ЭВМ 7
1.2. Структурная схема ЭВМ 9
1.3. Структура и функционирование управляемой оперативной памяти 10
1.4. Структура и функционирование интерпретирующего диалогового процессора 11
1.5. Структура и функционирование интерфейса ввода — вывода . . . 13
1.6. Система физического ввода — вывода информации 14

2. АРХИТЕКТУРА И ФУНКЦИИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ЭВМ „ИСКРА-226" 16

2.1. Состав программного обеспечения 16
2.2. Системное программное обеспечение 16
2.3. Сервисные средства 19
2.4. Обслуживающие средства 22
2.5. Программное обеспечение пользователя 24
2.6. Параллельное использование ресурсов ЭВМ 24

3. РАБОТА С МАГНИТНЫМИ НАКОПИТЕЛЯМИ 25

3.1. Инициализация дисков 26
3.2. Режим каталога файлов 26
3.3. Режим адресации секторов 28
3.4. Временные рабочие файлы на дисках 29

4. ОСНОВЫ ПРОГРАММИРОВАНИЯ НА ЯЗЫКЕ БЕЙСИК 29

4.1. Общие сведения 29
4.2. Простейшие конструкции языка и его возможности 30
4.3. Порядок подготовки и выполнения программ 33
4.4. Основные операторы 34
4.5. Операторы преобразования информации и функции символьных переменных 42

4.6. Операторы преобразования данных 43
4.7. Операторы ввода — вывода данных 43
4.8. Языки программирования БЕЙСИК-1 и БЕЙСИК-2 56
4.9. Расширение возможностей выразительных средств языка БЕЙСИК 58

5. МЕТОДИКА ПОДГОТОВКИ И РЕШЕНИЯ ЗАДАЧ НА ЭВМ „ИСКРА-226" 85

5.1. Задача Улама 85
5.2. Задача Штейнгауза 94
5.3. Задача „Взлеты и падения чисел-градин" 95
5.4. Моделирование параллельных динамических систем 96

6. НЕКОТОРЫЕ ОСОБЕННОСТИ РАБОТЫ С ЭВМ „ИСКРА-226" 103

6.1. Каталогизация файлов 103
6.2. Разработка программного обеспечения 105

7. МОДЕЛИРОВАНИЕ ДИСКРЕТНЫХ ПАРАЛЛЕЛЬНЫХ ДИНАМИЧЕСКИХ СИСТЕМ НА ЭВМ „ИСКРА-226" 134

7.1. Общие сведения 134
7.2. Использование локальных функций перехода 134
7.3. Использование интерактивной программной системы 138

ЗАКЛЮЧЕНИЕ 141

ЛИТЕРАТУРА 146

Информ. бланк № 103
Сдано в набор 23. 02. 88. Подписано в печать 20. 09. 88.
БФ 02709. Формат 84х108 1/32. Бум. тип. № 2.
Гарнитура Цюрих. Печать высокая.
Усл. печ. л. 7,98. Уч.-изд. л. 9,13.
Тираж 40 000экз. Заказ 8-310. Цена 65 к.

Главная редакция
Украинской Советской Энциклопедии
имени М. П. Бажана,
252601, ГСП, Киев-30, ул. Ленина, 51

Напечатано с оригинала-макета, подготовленного в
Главной редакции
Украинской Советской Энциклопедии
имени М. П. Бажана,
на Киевской книжной фабрике,
252054, Киев-54, ул. Воровского, 24

Справочное издание

Аладьев Виктор Захарович
Мартыненко Ярослав Григорьевич
Шиленко Владимир Федорович

ПЕРСОНАЛЬНЫЙ КОМПЬЮТЕР „ИСКРА–226" АРХИТЕКТУРА И ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ

Справочное руководство

Под редакцией
В. Ф. Шиленко

Научный редактор В. Ф. Хмель
Художественный редактор О. Д. Васильева
Технический редактор С. Ф. Бархатова
Корректор С. Я. Гапонова

Цена 65 к.

ПЕРСОНАЛЬНЫЙ КОМПЬЮТЕР

ИСКРА 226

АРХИТЕКТУРА И ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ

Справочное руководство

Рассмотрены технические, программные средства и возможности персонального компьютера „Искра-226", описаны операторы языка БЕЙСИК и особенности их выполнения.
Много внимания уделено прикладным программам, используемым в операционной среде БЕЙСИКа, в том числе программа A – BASIC, позволяющая расширить его выразительные средства.
Материал иллюстрирован практическими примерами.
Для широкого круга читателей, желающих использовать в своей деятельности возможности программно-управляемых ЭВМ.