Л.Н. МАРКЕЛОВА

Эксплуатация программоуправляемой вычислительной машины „ИСКРА 226"

«Машиностроение»

Л.Н. МАРКЕЛОВА

Эксплуатация программоуправляемой вычислительной машины „ИСКРА 226"

Допущено Главным Управлением подготовки и повышения квалификации работников учета ЦСУ СССР в качестве учебного пособия для подготовки и повышения квалификации кадров по программированию на ПЭКВМ "Искра 226"

МОСКВА
«МАШИНОСТРОЕНИЕ»
1987

ББК 32.97
М25
УДК 681.3.06

Рецензент канд. техн. наук **В. А. Ефремов**

Маркелова Л. Н.

М25 Эксплуатация программоуправляемой вычислительной машины «Искра 226»: Учеб. пособие для подготовки и повышения квалификации кадров по программированию на ПЭКВМ «Искра 226». — М.: Машиностроение, 1987. — 224 с.: ил.

(В обл.): 35 к.

Рассмотрены эксплуатационно-технические характеристики машины, дано описание и правила использования операторов языка БЕЙСИК при составлении программ для ПЭКВМ «Искра 226».

Пособие может быть полезно инженерно-техническим работникам и студентам при изучении языка БЕЙСИК и программировании решаемых задач на ПЭКВМ «Искра 226»

М $\frac{2405000000\text{-}017}{038(01)\text{-}87}$ 17-87 **ББК 32.97**

ПРЕДИСЛОВИЕ

Вычислительная техника развивается чрезвычайно бурно, как ни одна другая отрасль промышленности. Любая производственная деятельность человека включает элемент обработки информации и принятия решения для дальнейших действий на основе полученных результатов обработки. Для принятия правильного решения необходимо иметь достаточный объем сведений и уметь быстро воспользоваться этими сведениями. Основное назначение вычислительной техники состоит именно в том, чтобы ускорить темпы и качество обработки информации, т. е. в конечном итоге повысить производительность труда человека.

Вычислительная техника все больше внедряется в сферу производственной и научной деятельности человека, а также в сферу управления.

Любая техника для эффективного использования требует специальных знаний. При работе на ЭВМ пользователю необходимо в достаточном объеме иметь знания в области алгоритмизации и программирования. Поэтому вопросам подготовки кадров для работы с ЭВМ в настоящее время уделяется повышенное внимание.

Особенно удобны для оперативной работы в цехах, лабораториях, научных учреждениях мини-ЭВМ, которые в настоящее время вытесняют универсальные ЭВМ средней производительности.

Широко применяются мини-ЭВМ «Искра 1256» и «Искра 226», а также созданное на базе мини-ЭВМ «Искра 226» автоматизированное рабочее место технолога-программиста (АРМТП). Для диалоговых машин используется алгоритмический язык высокого уровня БЕЙСИК.

Книга знакомит читателя с вопросами использования языка БЕЙСИК при работе с мини-ЭВМ «Искра 226» и с вопросами операторской работы по эксплуатации машины.

УСЛОВНЫЕ СОКРАЩЕНИЯ

АПД — аппаратура передачи данных
АЦП — аналого-цифровой преобразователь
АРМТП — автоматизированное рабочее место технолога-программиста
БОСГИ — блок отображения символьно-графической информации
БИФ — блок интерфейсный функциональный
БФР — блок фильтра-распределителя
ВМО — внутреннее математическое обеспечение
ГМД — гибкий магнитный диск
ГП — графопостроитель
ИРПР — интерфейс для радиального подключения устройств с параллельной передачей информации
КНМЛ — кассетный накопитель на магнитной ленте
КП — канальный процессор
КС — контрольная сумма
МЛ — магнитная лента
НГМД — накопитель на гибком магнитном диске
НМД — накопитель на жестком магнитном диске
НМЛ — накопитель на магнитной ленте
НС — непосредственный счет
ОП — оперативная память
ОУП — оперативная управляющая память
ПЗУ — постоянное запоминающее устройство
ПИД — процессор интерпретирующий диалоговый
ПУ — печатающее устройство
ПЭКВМ — программоуправляемая электронная клавишная вычислительная машина
СФ — спецфункция
УВВ — устройство ввода-вывода
УГИ — указатель графической информации
УК — устройство клавишное
ФАУ — физический адрес устройства
ЦАП — цифроаналоговый преобразователь
ЦП — центральный процессор
ЭВМ — электронная вычислительная машина
ЭЛТ — электронная лучевая трубка

Глава I

ОБЩИЕ СВЕДЕНИЯ О ПЭКВМ «ИСКРА 226»

1. НАЗНАЧЕНИЕ И ОБЛАСТЬ ПРИМЕНЕНИЯ

Программоуправляемая электронная клавишная вычислительная машина «Искра 226» (в дальнейшем — ПЭКВМ «Искра 226») относится к классу мини-ЭВМ. Как любая другая машина, она нуждается в управлении при выполнении возложенных на нее задач. Такое управление осуществляется с помощью программ. Пока программа не подготовлена и не размещена в памяти машины, машина «не знает», что она должна делать, какие данные вводить, какие действия выполнять. Таковы особенности всех вычислительных программоуправляемых машин, какими бы совершенными они не были.

«Искра 226» — машина клавишная. Это значит, что пользователь для «общения» с машиной использует клавиатуру, с помощью которой он может вводить в машину программу и данные, а также управлять ее работой.

Машина обладает достаточной гибкостью языковых и технических средств, поэтому ее можно отнести к универсальным ЭВМ, причем действия машины могут быть запрограммированы для решения различных проблем и задач.

ПЭКВМ «Искра 226» выпускается с различной комплектностью устройств. В зависимости от исполнения машина предназначена для:

выполнения в диалоговом режиме оперативных плановых расчетов;

решения в диалоговом режиме экономических, статистических, инженерно-технических задач и задач оптимизации;

решения инженерных, научно-технических задач и обработки исследовательской информации при непосредственном участии пользователя (инженера, научного работника, исследователя) в процессе обработки;

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

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

автоматизации трудоемких расчетов при выполнении исследовательских, проектных, конструкторских, инженерных работ.

Основными областями применения ПЭКВМ «Искра 226» являются:

отраслевые и академические институты;

проектные и конструкторские организации;

промышленные, строительные, автотранспортные предприятия и управления;

высшие и средние учебные заведения;

вычислительные центры;

вычислительная система ЦСУ СССР;

отделы и организации Госплана СССР, Госснаба СССР и др.; лечебно-профилактические учреждения.

2. ТЕХНИКО-ЭКСПЛУАТАЦИОННЫЕ ДАННЫЕ ПЭКВМ «ИСКРА 226»

ПЭКВМ «Искра 226» — машина диалоговая, программоуправляемая. Входной язык — БЕЙСИК.

Объектами обработки являются числа и текст (символьные данные). Числа при вводе-выводе представляются как целые или действительные в естественной и экспоненциальной формах, максимальная разрядность — 13 десятичных разрядов. Диапазон представления порядка чисел в экспоненциальной форме составляет от — 99 до + 99.

Автоматически в машине выполняются арифметические операции и операции сравнения, вычисляются значения элементарных функций и функций символьного переменного, выполняются операторы языка БЕЙСИК.

Техническая характеристика ПЭКВМ «Искра 226»

Объем оперативной памяти (ОП), байт	64К
Объем оперативной управляющей памяти (ОУП), байт	64К
Объем постоянной управляющей памяти (ПЗУ), байт	16К
Среднее время выполнения операций, с, не более:	
арифметических	0,001
извлечения квадратного корня	0,002
вычисления значений элементарных функций	0,05

ПЭКВМ «Искра 226» позволяет выполнять:

ввод программы и данных с клавиатуры или технического носителя (магнитной ленты или магнитного диска) в ОП;

редактирование программы;

вывод программы и результатов вычислений на экран электронно-лучевой трубки (ЭЛТ), печатающее устройство, на технический носитель;

сопряжение с аппаратурой передачи данных (АПД) по рангу С2 машин ЕС ЭВМ и СМ ЭВМ;

вывод графической информации на экран и графопостроитель (исполнения 3, 4, 6);

обмен информацией с СМ ЭВМ по рангу ИРПР (исполнения 3, 6);

обмен информацией с цифровыми приборами и аналоговыми устройствами (исполнение 6);

ввод графической информации с указателя графической информации (УГИ) (исполнение 6);

ввод в ОУП и вывод системного (внутреннего) математического обеспечения (ВМО) с магнитного диска (кроме исполнения 5);

интерпретативную реализацию операторов программы пользователя, записанной в ОП.

Функциональные возможности ПЭКВМ «Искра 226» реализуются аппаратно, аппаратно-программно и программно.

3. СОСТАВ ИСПОЛНЕНИЙ И ХАРАКТЕРИСТИКА СОСТАВНЫХ УСТРОЙСТВ

Состав ПЭКВМ «Искра 226». В комплект машины входят:

процессор интерпретирующий диалоговый (ПИД) с устройством клавишным (УК);

печатающее устройство (ПУ);

набор стандартных для ПЭКВМ устройств ввода-вывода (УВВ) и соответствующие им блоки интерфейсные функциональные (БИФ) для подключения УВВ к ПИД;

набор БИФ для подключения нестандартного оборудования.

Машина выпускается в нескольких исполнениях, различающихся набором УВВ и БИФ (табл. 1).

Назначение и состав ПИД. ПИД предназначен для обработки информации по программе пользователя, записанной в оперативной памяти, и управления устройствами ввода-вывода.

В состав ПИД входят:

центральный процессор (ЦП), управляющий работой всей машины и выполняющий операции арифметико-логического преобразования информации;

канальный процессор (КП), управляющий обменом информацией между оперативной памятью и УВВ по заданию, полученному от центрального процессора;

блок оперативной памяти (ОП), обеспечивающий хранение программы пользователя, обрабатываемых по программе данных, результатов обработки, а также служебной информации (адресов возврата из подпрограмм, управляющей информации для подпрограмм и циклов, информации об операндах, информации о данных); условно в ОП можно выделить зону программы, зону данных, служебную и свободную зоны;

блоки оперативной управляющей памяти (ОУП), обеспечивающие хранение загружаемого внутреннего математического обеспечения (ВМО);

блоки постоянной управляющей памяти (ПЗУ), содержащие встроенное математическое обеспечение;

блок отображения символьно-графической информации (БОСГИ) индикатор ПИД, обеспечивающий индикацию на экране электронно-лучевой трубки (ЭЛТ) символьной информации в 24 строки по 80 символов и графической информации, максимальный объем которой составляет 256·560 точек;

кассетный накопитель на магнитной ленте (КНМЛ) — встроенное устройство ввода-вывода информации на магнитную ленту в мини-кассете (КМЛ); КНМЛ входит в ПИД лишь в исполнениях 5 и 6 ПЭКВМ «Искра 226».

ПИД выполнен в отдельном корпусе (рис. 1). Расширитель ввода-вывода ПИД имеет семь гнезд для подключения БИФ. Одно гнездо за-

1. Таблица комплектности машин по исполнению

Наименование	Тип УВВ	Тип БИФ	Исполнения ПЭКВМ «Искра 226»					
			1	2	3	4	5	6
ПИД	Исполнение 1 Исполнение 2 (с КНМЛ)	—	1 —	1 —	1 —	1 —	— 1	— 1
ПУ	РОБОТРОН 1154 РОБОТРОН 1156 ДЗМ 180	015-30-01 015-31-01 015-33	1	1	1	1	1	1
НГМД	ЕС-5074 (005-51) PL45 (005-50)	015-21	1	1	1	1	—	1
НМД	ИЗОТ-1370 (005-70) СМ-5400 (005-71)	015-23	—	1	1	—	—	—
НМЛ	СМ-5300 (005-61) СМ-5003 (005-60)	015-25	—	—	1	—	—	—
ГП	Н-306	015-13	—	—	1	1	—	—
УГИ	007-50	015-60	2	—	—	—	—	1
ЦАП	—	015-10	—	—	—	—	—	1
АЦП	—	015-14	—	—	—	—	—	1
Блок связи с ЭВМ по ИРПР	—	015-82	—	—	1	—	—	1
Блок связи с прибором	—	015-83	—	—	—	—	—	1
Блок связи с ЭВМ через АПД по стыку С2	—	015-85	1	1	1	1	1	1

нимает БИФ, через который подключаются к процессору клавишное и печатающие устройства.

Рис. 1. Процессор интерпретирующий диалоговый (ПИД)

Диалоговый режим работы ПИД (диалог пользователя-оператора и машины) обеспечивается консольными устройствами ввода-вывода (УК и БОСГИ).

Назначение БОСГИ и правила индикации. БОСГИ предназначен для индикации:

диалоговой информации, т. е. информации, набираемой оператором-пользователем на клавиатуре, а также информации о состоянии блоков и устройств машины и о состоянии выполняемой программы пользователя, выдаваемой процессором; при такой индикации БОСГИ выполняет функцию консольного устройства вывода — CONSOL OUTPUT;

информации, выводимой операторами печати; БОСГИ при этом является устройством вывода индицируемой алфавитно-цифровой информации (программы и данных из ОП);

графической информации, при этом БОСГИ выполняет функции графического устройства вывода.

Индикация информации любого типа на экране осуществляется в негативном изображении, т. е. светлыми элементами на темном фоне, однако возможно задание позитивного изображения для символьного БОСГИ.

БОСГИ можно использовать и как графический, и как символьный индикатор параллельно, т. е. одновременно на экране может находиться и графическая, и символьная информация.

Указателем позиции (знакоместа) на экране для вывода нового символа служит символьный курсор (символ «подчеркивание»), а для вывода новой точки графического изображения — графический курсор (уголок с вершиной в точке «г»).

Физическим началом символьного экрана является первая позиция первой (верхней) строки экрана, физическим началом координат графического экрана — левый нижний угол.

Физическая строка экрана составляет 80 позиций, что определяется

конструкцией БОСГИ, а строка печати задается программно оператором SELECT.

Существует также понятие логической строки — это строка в объеме информации, набираемой на клавиатуре. Максимальная длина ее определяется объемом буфера ввода — 240 символов. Логическая строка всегда начинается с начала физической строки экрана индикацией служебного символа «:» («двоеточие») или «*» («звездочка») и занимает не более трех физических строк экрана.

Переход курсора в начало следующей строки выполняется автоматически после заполнения предыдущей. Перед переходом из последней строки экрана изображение всех строк перемещается вверх на одну строку, при этом изображение первой физической строки экрана теряется, последняя строка экрана освобождается и курсор переходит в ее первую позицию. Сочетания некоторых служебных символов на экране, индицируемых машиной, имеют в диалоге следующий смысл:

символы «двоеточие» и «курсор» (:—) в начале строки означают, что машина не выполняет никаких действий, но готова к их выполнению и ждет управляющей информации с клавиатуры;

символы «вопрос» и «курсор» (?—) в двух последних позициях логической строки означают ожидание набора данных, без которых машина не может продолжать выполнение программы;

символ «двоеточие» или «звездочка» в начале строки и «курсор» в любой позиции логической строки означают ожидание набора очередного символа или завершения набора (:PRINT"ТАБЛ_).

Устройство клавишное (УК). Назначение клавиш. Клавиатура является консольным устройством ввода — CONSOL INPUT, но может использоваться и как периферийное устройство ввода.

УК предназначено для ввода символов, изображение которых нанесено на клавиши, т. е. для формирования и передачи в процессор кодов этих символов (табл. 2).

По назначению клавиш на УК можно выделить восемь зон и несколько специальных клавиш: SHIFT, SHIFT LOCK, STMT NUMBER, CR/LF; переключатель РУС/ЛАТ; кнопку RESET (рис. 2).

Первая зона — телетайпная клавиатура, работающая в трех регистрах. В нижнем регистре набираются операторы языка БЕЙСИК, а в верхнем в зависимости от положения переключателя РУС/ЛАТ — буквы русского алфавита и цифры или буквы латинского алфавита и знаки. Исключение составляет клавиша «Пробел», она не зависит от положения регистров, т. е. в любом регистре нажатие на эту клавишу означает пробел. При включении ПИД автоматически устанавливается верхний регистр УК. Лампочка индикации между клавишами SHIFT и SHIFT LOCK погашена.

Нижний регистр устанавливается нажатием либо клавиши SHIFT, либо клавиши SHIFT LOCK (с фиксацией нижнего регистра). При отжатии клавиши SHIFT автоматически устанавливается верхний регистр, поэтому для восстановления верхнего регистра нужно кратковременно нажать на клавишу SHIFT.

2. Коды клавиш

Наименование клавиш	Шестнадцатеричный код клавиши			
	Переключатель РУС		Переключатель ЛАТ	
	Нижний регистр	Верхний регистр	Нижний регистр	Верхний регистр
0	30	30	30	30
1	31	31	31	31
2	32	32	32	32
3	33	33	33	33
4	34	34	34	34
5	35	35	35	35
6	36	36	36	36
7	37	37	37	37
8	38	38	38	38
9	39	39	39	39
»	2E	2E	2E	2E
EXP (—	2D	2D	2D	2D
SQR (+	2B	2B	2B	2B
LOG (/	2F	2F	2F	2F
ABS (*	2A	2A	2A	2A
Пробел	20	20	20	20
; +	2B	3B	2B	3B
1 !	21	31	21	31
2 "	22	32	22	32
3 #	23	33	23	33
4 ¤	24	34	24	34
5 %	25	35	25	35
6 &	26	36	26	36
7 '	27	37	27	37
8 (	28	38	28	38
9)	29	39	29	39
0 :	3A	30	3A	30
— =	3D	2D	3D	2D
: *	2A	3A	2A	3A
.	3E	2E	3E	2E
,	3C	2C	3C	2C
/ ?	3F	2F	3F	2F
ПИ —	DF	DF	2D	2D
STMT NUMBER	81	91	A1	B1
RUN	82	82	82	82
CLEAR	83	93	A3	B3
LIST	84	94	A4	B4
PRINT	86	96	A6	B6
LOAD	87	97	A7	B7
SAVE	88	98	A8	B8

Продолжение табл. 2

Наименование клавиш	Шестнадцатеричный код клавиши			
	Переключатель РУС		Переключатель ЛАТ	
	Нижний регистр	Верхний регистр	Нижний регистр	Верхний регистр
ARC	2C	2C	2C	2C
TAN (;	3B	3B	3B	3B
SIN ((	28	28	28	28
COS ( )	29	29	29	29
Й J	CA	EA	6A	4A
Ц C	C3	E3	63	43
У U	D5	F5	75	55
К K	CB	EB	6B	4B
Е E	C5	E5	65	45
Н N	CE	EE	6E	4E
Г G	C7	E7	67	47
Ш |	DB	FB	7B	58
Щ]	DD	FD	7	5
З /	DA	FA	7A	5A
Х H	C8	E8	68	48
Ф F	C6	E6	66	46
Ы Y	D9	F9	79	59
В W	D7	F7	77	57
А A	C1	E1	61	41
П P	D0	F0	70	50
Р R	D2	F2	72	52
О O	CF	EF	6F	4F
Л L	CC	EC	6C	4C
Д D	C4	E4	64	44
Ж V	D6	F6	76	56
Э	D6	FC	7C	5C
Ч Λ	DE	FE	7E	5E
С S	D3	F3	73	53
М M	CD	ED	6D	4D
И I	C9	E9	69	49
Т T	D4	F4	74	54
Ь X	D8	F8	78	58
Б B	C2	E2	62	42
Ю @	C0	E0	40	40
Я Q	D1	FA	71	51
LINE EPASE	9F	9F	9F	9F
CONTINUE	8A	8A	8A	8A
BACKSPACE	89	89	89	89
DELETE	9D	9D	9D	9D

Продолжение табл. 2

Наименование клавиш	Шестнадцатеричный код клавиши			
	Переключатель РУС		Переключатель ЛАТ	
	Нижний регистр	Верхний регистр	Нижний регистр	Верхний регистр
RECALL	9B	9B	9B	9B
CR/LF	85	85	85	85
INSERT	9C	9C	9C	9C
ERASE	9E	9E	9E	9E
EDIT	9A	9A	9A	9A
0 вверх	AA	AA	AA	AA
1 ←—	AB	AB	AB	AB
2 ←	AC	AC	AC	AC
3 →	AD	AD	AD	AD
— — →	AE	AE	AE	AE
— вниз	AF	AF	AF	AF

Нижнее положение переключателя РУС/ЛАТ соответствует латинскому регистру при погашенной лампочке индикации.

STMT NUMBER, клавиша верхняя справа, используется независимо от положения регистров для автоматического набора очередного номера строки программы, вводимой с УК. Номера устанавливаются с шагом 10.

CR/LF (возврат каретки и перевод строки) играет роль пусковой клавиши в диалоге. Нажатием этой клавиши завершается набор диалоговой информации, т. е. строки программы, данного или группы данных, информации редактирования.

При наборе операторов языка БЕЙСИК можно пользоваться и буквами латинского алфавита, но нельзя пользоваться буквами русского алфавита, имеющими одинаковое начертание с буквами латинского алфавита.

Вторая зона содержит следующие клавиши редактирования набираемой информации до нажатия клавиши CR/LF:

BACK SPACE — стирание символа перед курсором с перемещением курсора на место стертого символа;

LINE ERASE — стирание всей логической строки.

Вторая зона содержит также клавиши управления счетом:

RUN — пуск программы на счет;

HALT/STEP — останов или пошаговое исполнение программы;

CONTINUE — продолжение счета.

Третья зона — цифровая клавиатура (содержит цифры, десятичную точку). Коды одинаковых цифр первой и третьей зон совпадают.

Четвертая зона содержит клавиши с операторами языка БЕЙСИК, часто используемыми в диалоге;

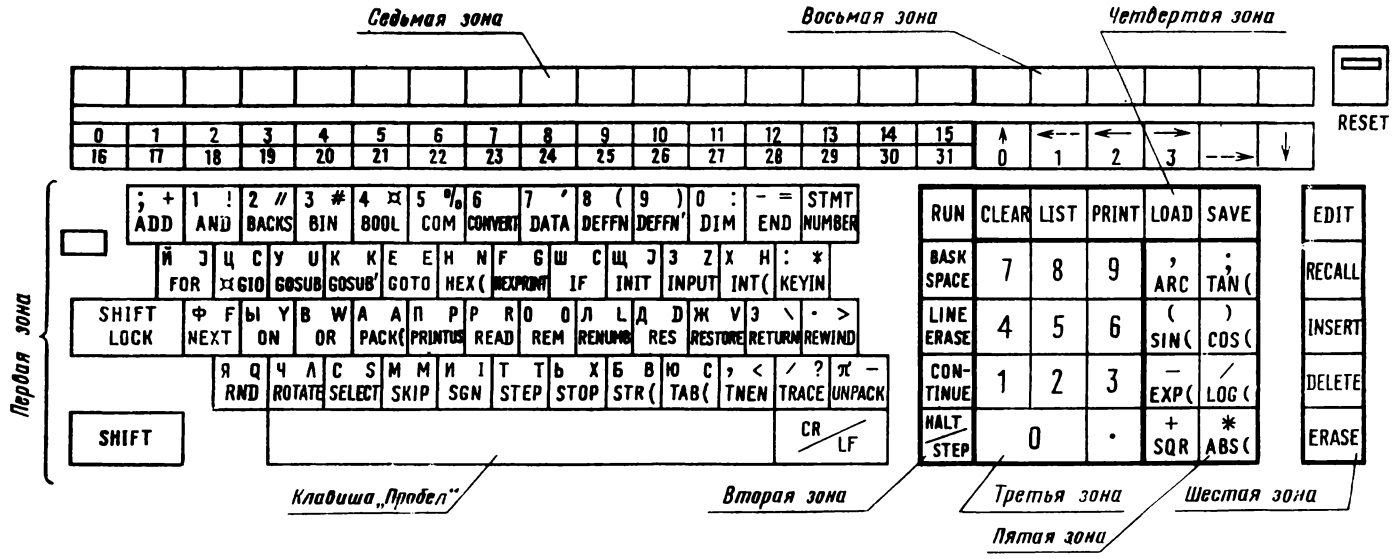


Рис. 2. Устройство клавишное (УК)

CLEAR — оператор очистки ОП;

LIST — оператор индикации программы и справочной информации;

PRINT — оператор печати и индикации данных;

LOAD, SAVE — операторы ввода-вывода информации с диска и магнитной ленты.

Пятая зона содержит клавиши со знаками арифметических операций: +, —, /, *; стандартных математических функций: LN (,SIN (,...; разделителей:,; ( ). Используются клавиши в двух регистрах.

Шестая зона содержит клавиши редактирования:

EDIT — задание режима редактирования текста строки программы;

RECALL — вызов редактируемой строки программы на экран;

INSERT — раздвижка текста для вставки символа над курсором;

DELETE — стирание символа над курсором с подтягиванием оставшихся символов логической строки на одну позицию;

ERASE — стирание части логической строки, начиная с символа над курсором.

Седьмая зона — клавиши спецфункций (СФ), имеющие нумерацию в нижнем регистре 0—15, в верхнем регистре 16—31.

Клавиши СФ не имеют фиксированных значений. Пользователь может программно присвоить этим клавишам свои значения.

Восьмая зона содержит клавиши управления курсором, используемые при редактировании для перемещения курсора в пределах логической строки экрана:

← (→) — перемещение в предыдущую (следующую) позицию логической строки экрана;

←—— (——→) — перемещение на пять позиций назад (вперед);

↑ (↓) — перемещение в позицию с тем же номером предыдущей (следующей) физической строки экрана.

Редактируемая логическая строка начинается с символа «*».

RESET — кнопка аварийного останова со стиранием экрана, в режиме ожидания вызывает лишь стирание экрана, а программа и данные в памяти сохраняются.

Печатающее устройство (ПУ). В состав каждого исполнения входит ПУ с матричным типом печати, предназначенное для вывода на печать алфавитно-цифровой и символьной информации. К таким устройствам относятся: ДЗМ 180 (158 знаков в строке, средняя скорость печати не менее 40 полных строк в минуту или 180 знаков в секунду); Роботрон 1154 (до 132 знаков в строке, скорость печати до 50 знаков в секунду); Роботрон 1156 (до 178 знаков в строке, скорость печати до 100 знаков в секунду).

Для печати используется красящая лента шириной 13 мм. Каждому ПУ соответствует свой БИФ.

Дисковые запоминающие устройства. Дисковые устройства предназначены для записи на гибкий (ГМД) или жесткий (МД) магнитный диск информации, выводимой из оперативной памяти, а также для считывания информации и ввода ее в оперативную память. В состав ПЭКВМ «Искра 226» может входить устройство, работающее с гибки-

ми магнитными дисками (в дальнейшем ГМД или просто дисками), «Искра 005-51» или «Искра 005-50»).

Устройство «Искра 005-51» (исполнение 1) состоит из двух накопителей (нижнего и верхнего) на гибком магнитном диске (НГМД) ЕС 5074, расположенных в одном корпусе. Каждый из двух НГМД ЕС 5074 имеет одну магнитную головку и использует для работы одну рабочую поверхность диска емкостью 250 К байт (1001 сектор по 256 байт), т. е. одновременно для работы можно устанавливать два диска. Скорость обмена с НГМД составляет 250 К бит/с.

Устройство «Искра 005-50» выполнено на базе НГМД PL×45D, имеет две магнитные головки и использует четыре рабочие поверхности двух устанавливаемых в НГМД дисков. При работе для каждого диска возможен доступ лишь к секторам одной поверхности диска (со стороны магнитной головки). Для обеспечения доступа к секторам другой поверхности диска нужно ее сменить вручную, т. е. вынуть диск и установить его обращенным другой поверхностью к магнитной головке. Емкость одной рабочей поверхности 250 К байт. Скорость обмена с НГМД 250 К бит/с. Подключается любое из названных устройств к процессору через БИФ «Искра 015-21».

Накопитель на жестких магнитных дисках (НМД) «Искра 005-71» (на базе СМ-5400) или «Искра 005-70» (НМД ИЗОТ 1370) входит в состав ПЭКВМ «Искра 226» исполнений 2 и 3. Накопитель имеет четыре магнитные головки и соответственно четыре рабочие поверхности на двух дисках: фиксированном нижнем и съемном верхнем. Каждая рабочая поверхность имеет информационную емкость 1224К байт (4896 секторов по 256 байт). Частота вращения диска 2400 об/мин обеспечивает скорость обмена 2,5 Мбит/с. Работа НМД возможна при установке обоих дисков. Подключение к процессору через БИФ «Искра 015-23».

Накопитель на магнитной ленте. В состав ПЭКВМ «Искра 226» (исполнение 3) в качестве внешнего накопителя информации с записью на магнитную девятидорожечную ленту входит одно из устройств: «Искра 005-60» — НМЛ ИЗОТ 5003 или «Искра 005-61» — НМЛ СМ 5003. Устройства выполняют запись информации блоками переменной длины (от 18 до 1024 байт) с плотностью записи 32 бит на 1 мм. Длина межблочных промежутков переменная — от 12,7 до 7600 мм. Вся информация записывается между маркерами физического начала и физического конца магнитной ленты. Скорость обмена до 10 К байт/с. Информация, записанная указанными устройствами на магнитную ленту, может быть использована в ЕС ЭВМ. Подключается НМЛ к процессору через БИФ «Искра 015-25».

Графопостроитель Н-306. Графопостроитель Н-306 предназначен для графической реализации зависимостей $Y \quad f(x)$ и $Y \quad f(t)$ на планшете с размерами рабочего поля 30 см по каналу «X» и 20 см по каналу «Y». Крепление планшета осуществляется электростатическим полем. Возможны установка начального положения пера в любой точке рабочего поля и ручное регулирование масштаба регистрации. Максимальная скорость регистрации не менее 75 см/с. Заправляется пи-

шущее устройство только специальными чернилами. Подключается графопостроитель через БИФ «Искра 015-13».

Указатель графической информации. В состав исполнения 6 машины входит устройство «Искра 007-50» — указатель графической информации (УГИ). УГИ предназначен для ввода в ПИД преобразованных в двоичный код значений координат X и Y углового перемещения ручки-манипулятора. Исходное положение ручки вертикальное (начало координат), перемещение ручки дискретное. На каждой полуоси имеется по восемь дискретных значений координат. В горизонтальной плоскости перемещение ручки-манипулятора УГИ осуществляется без ограничений при отклонении от вертикального положения в диапазоне $\pm$ 60°. Подключается УГИ через БИФ «Искра 015-60».

Нестандартное оборудование. Кроме перечисленных устройств к ПЭКВМ «Искра 226» могут быть подключены аналоговые и цифровые приборы и устройства, не входящие в состав УВВ машины, а также ЕС ЭВМ и СМ ЭВМ. Подключаемое устройство может быть источником информации, приемником информации или приемоисточником. Подключение возможно при наличии соответствующего БИФ.

БИФ телекоммуникационный «Искра 015-85» предназначен для подключения ПИД через аппаратуру передачи данных (АПД) к средствам передачи и телеобработки ЕС и СМ ЭВМ. БИФ обеспечивает обмен информацией с АПД по стыку С2 (ГОСТ 18145-81) без автоматического вызова устройства. Информация обмена записывается в коде КОИ-8. Скорость обмена старт-стопного составляет 100, 200, 300, 600, 1200, 2400 бит/с, синхронного — не более 2400 бит/с.

Для связи используются телефонные линии связи. БИФ выполняет управление АПД как в дуплексном режиме (когда прием и передача ведутся одновременно), так и в полудуплексном (прием или передача). БИФ выполняет требования протокола, принятого в терминалах ЕС 7920 при синхронном обмене и в терминалах ЕС 8570 при старт-стопном обмене.

В каждом комплекте машины поставляется БИФ «Искра 015-85», а в комплекте документации — библиотека прикладных и загружаемых программ, реализующая обмен в старт-стопном режиме со скоростью 100 бит/с (в режиме эмуляции ЕС 8570).

БИФ «Искра 015-82» предназначен для обмена информацией между ПЭКВМ «Искра 226» (исполнения 3 и 6) и СМ ЭВМ по рангу ИРПР (в соответствии с работой интерфейса для радиального подключения устройств с параллельной передачей информации). БИФ выполняет передачу информации в машины СМ-3 и СМ-4 и передачу в ПИД информации, принятой от СМ-3 или СМ-4. Соединение машин кабельное.

БИФ «Искра 015-83» предназначен для подключения к ПИД цифровых и регистрирующих приборов по приборному интерфейсу в соответствии с ГОСТ 26.003-80 и обеспечивает работу в режимах системного контроллера.

БИФ «Искра 015-10» — цифроаналоговый преобразователь (ЦАП), используется для подключения к машине аналогового прибора—

приемника информации. Он преобразует выводимое число в электрический сигнал, в одном из диапазонов: ±5 В, ± 0,5 В. Время преобразования числа не более 100 мкс.

БИФ «Искра 015-14» — многоканальный аналого-цифровой преобразователь (АЦП), используется для подключения приборов — источников информации. Он преобразует напряжение входного электрического сигнала в девятиразрядное двоичное число, т. е. АЦП является цифровым измерителем напряжения. Время преобразования не более 50 мкс. АЦП работает в режимах: однопроводном, дифференциальном, циклическом, ждущем. АЦП имеет 32 однопроводных канала преобразования, 16 дифференциальных каналов преобразования.

Адресация устройств ввода-вывода. В машине все устройства ввода-вывода (УВВ) и соответствующие им блоки интерфейсные функциональные (БИФ), а также блоки интерфейсные функциональные, используемые для подключения нестандартных устройств, закодированы двухразрядными шестнадцатеричными числами, которые называются физическими адресами устройств (ФАУ).

Физические адреса присваиваются конструктивно заводом-изготовителем. Физический адрес устройства используется при программировании обмена информацией процессора с устройствами ввода-вывода (табл. 3).

При включении машины считаются автоматически заданными для программного обращения адреса следующих устройств: устройства клавишного, символьного БОСГИ, графического БОСГИ, НГМД, КНМЛ.

3. Физические адреса устройств (ФАУ) ввода-вывода

Тип УВВ	ФАУ шестнадцатеричный
Устройство клавишное (УК)	01
Символьный БОСГИ	05
Накопитель на кассетной магнитной ленте (КНМЛ)	08
Печатающее устройство (ПУ)	0C
Графический БОСГИ	10
Графопостроитель Н-306	14
Накопитель на гибком магнитном диске (НГМД)	18
Накопитель на магнитной ленте (НМЛ-9)	1B
Накопитель на магнитном диске (НМД)	1C
Аналого-цифровой преобразователь (АЦП)	20
То же	21
»	22
»	23
Цифроаналоговый преобразователь (ЦАП)	24
То же	25
»	26
Указатель графической информации (УГИ)	27
Интерфейс для радиального подключения устройств с параллельной передачей информации (ИРПР)	2D
Приборный интерфейс	2E
Передатчик по рангу С2	34
Приемник по рангу С2	35

4. МАТЕМАТИЧЕСКОЕ ОБЕСПЕЧЕНИЕ МАШИНЫ

В поставляемом с машиной математическом (программном) обеспечении можно выделить следующие основные части:

системное математическое обеспечение, которое в свою очередь состоит из совокупности микропрограмм в ПЗУ — встроенного математического обеспечения и внутреннего математического обеспечения (ВМО), загружаемого в ОУП с НГМД или КНМЛ;

библиотека прикладных программ.

Встроенное математическое обеспечение. Встроенное математическое обеспечение выполняет программное управление работой машины сразу же при ее включении, задавая особый режим работы машины, который условно можно назвать режимом ЗАГРУЗЧИКА. В этом режиме машина обеспечивает:

загрузку ВМО, т. е. ввод в ОУП программ внутреннего математического обеспечения с НГМД или КНМЛ;

диагностику основных устройств машины;

ввод с УК, редактирование и вывод на технический носитель программ ВМО;

отладочный счет по программе ВМО.

Пользователь практически выполняет лишь загрузку ВМО и диагностику устройств машины. Остальные режимы используются «системным» программистом при создании ВМО. Команды, набираемые на УК, содержат операторы языка загрузчика. Назначение и правила применения языка подробно изложены в эксплуатационной документации машины.

Внутреннее математическое обеспечение. ВМО обеспечивает применение языка БЕЙСИК в качестве входного языка программирования, а также наиболее полное использование ресурсов машины в процессе эксплуатации за счет параллельной работы устройств ввода-вывода и процессора машины. Работу машины под управлением ВМО можно назвать «эксплуатационным режимом», т. е. режимом работы, при котором выполняется программа пользователя.

ВМО состоит из следующих функциональных блоков: начальной генерации, управления задачей, транслятора, интерпретатора, ретранслятора, управления заданиями на ввод-вывод, обработки прерываний, реализации заданий на ввод-вывод, обработки ошибок. Основным аппаратом записи и кодирования ВМО является система инструкций машины.

Глава II
ПОДГОТОВКА МАШИНЫ К РАБОТЕ

1. ВКЛЮЧЕНИЕ ПЭКВМ «ИСКРА 226»

ПИД и все устройства ввода-вывода должны быть подключены к блоку фильтра-распределителя (БФР) «Искра 020-01», обеспечивающего повышенную помехозащищенность при включении в сеть. Порядок включения машины следующий:

1) включить в сеть трехполюсной вилкой блок фильтра-распределителя;

2) включить ПИД тумблером на левой боковой панели корпуса ПИД. Включение сопровождается звуковым сигналом, шумом включаемых двигателей вентиляторов и через 15—20 с появлением на экране информации о включении в виде:

```
ЗАГРУЗЧИК
*_
```

3) включить устройства ввода-вывода, необходимые для работы, установкой переключателей сети (питания) в положение ВКЛ.

Если при включении ПИД на экране появилась информация, отличная от указанной, или она вообще не появилась, необходимо выключить ПИД тумблером, а затем, выждав паузу в 30—40 с, включить снова.

Включение машины выполняется в обратном порядке, т. е. сначала необходимо выключить все УВВ, затем ПИД и блок фильтра-распределителя.

2. ПОДГОТОВКА НАКОПИТЕЛЕЙ НА ГИБКИХ МАГНИТНЫХ ДИСКАХ К РАБОТЕ

Устройство «Искра 005-51». Устройство «Искра 005-51» выполнено на базе двух накопителей ЕС 5074 (рис. 3). Оно имеет на передней панели корпуса две горизонтальные прорези с крышками для установки гибких дисков в НГМД: верхний, обозначаемый *R*, и нижний — *F*, а также по две лампочки *5* и *8* индикации на каждый НГМД.

Левая лампочка *8* загорается при обращении ПИД к НГМД и горит во время обмена. Правая лампочка *5* индицирует защиту записи. Защита записи означает, что с диска, установленного в данный НГМД, при наличии индикации можно лишь считывать информацию. Рядом

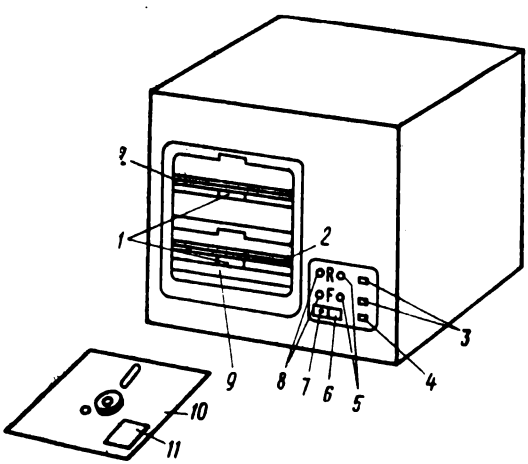


Рис. 3. Устройство «Искра 005-51» на базе НГМД ЕС 5074:

1 — клавиша фиксации дисковода; *2* — крышка дисковода; *3* — кнопка «Сброс защиты записи»; *4* — кнопка начальной установки; *5* — индикатор «Защита записи»; *6* — переключатель сети; *7* — индикатор включения сети; *8* — индикатор «Выбор диска»; *9* — прорезь для установки гибкого диска; *10* — гибкий диск; *11* — наклейка на диске

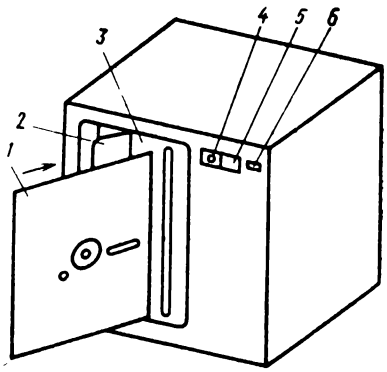


Рис. 4. Устройство «Искра 005-50» на базе PL×45D2:

1 — гибкий диск; *2* — дверца диска; *3* — дисковод; *4* — индикатор включения сети; *5* — переключатель сети; *6* — кнопка начальной установки

с лампочкой находится кнопка *3* без фиксации, предназначенная для снятия защиты записи. Под лампочками расположены переключатель *6* питания НГМД со встроенной лампочкой *7* индикации готовности НГМД к работе. Правее переключателя находится кнопка *4* (без фиксации) начальной установки и восстановления защиты записи на дисках *F* и *R* одновременно. При нажатии на эту кнопку устанавливается защита записи, о чем свидетельствует индикация обеих лампочек защиты записи.

Для *подготовки НГМД к работе* необходимо:

открыть крышку НГМД; крышка поднимается пружиной, и открывать ее следует плавно, не прикладывая усилия вверх; для открывания достаточно «сжать» крышку посередине (нажать на клавишу фиксации НГМД) и, удерживая, дать крышке подняться;

включить питание; при этом загорается лампочка на переключателе и появляется индикация защиты записи для *F* и *R* одновременно;

установить в НГМД *F* или НГМД *R* диск, который помещен в темный плотный конверт; при установке диска этикетка, наклеенная на конверте, должна находиться в ближнем правом углу сверху; устанавливать диск следует без перекоса до упора;

закрыть крышку, нажав на нее сверху (до щелчка);

снять кнопкой, соответствующей типу диска (*F* или *R*), защиту записи, если на диск будет записываться информация;

при необходимости аналогично установить диск во второй НГМД.

Для *замены диска* в НГМД необходимо:

открыть крышку НГМД (при открывании диск выталкивается, поэтому следует предупредить возможное падение диска);

установить диск замены так же, как при подготовке НГМД.

Установка и снятие диска выполняются при включенном НГМД.

Устройство «Искра 005-50» (PL×45D). Устройство имеет на передней панели корпуса (рис. 4) две вертикальные прорези для установки диска в дисководы: правый — *F*, левый — *R*. На панели находится также переключатель питания, лампочка индикации готовности устройства и кнопка начальной установки (справа от переключателя).

Для *подготовки НГМД к работе* необходимо:

включить питание переключателем; при этом загорается лампочка индикации;

нажать кнопку начальной установки; магнитная головка не более чем через 10 с переместится на предельную дорожку;

услышав щелчок, открыть дверцу дисковода *F* или *R*;

вставить диск до упора (рабочая поверхность диска справа);

закрыть дверцу дисковода.

Перед выключением питания диск обязательно вынуть. Защита записи в НГМД «Искра 005-50» не предусмотрена.

3. ПОДГОТОВКА НАКОПИТЕЛЯ НА ЖЕСТКИХ МАГНИТНЫХ ДИСКАХ К РАБОТЕ

НМД имеет на передней панели корпуса лампочку индикации и переключатель СЕТЬ, переключатель СТАРТ, используемый для включения и остановки двигателя НМД, лампочки индикации ГОТОВО и ЗАГРУЗКА; лампочки индикации, совмещенные с клавишами ЗАЩИТА и НЕИСПРАВНО (рис. 5).

Работать с НМД можно лишь при наличии двух дисков фиксированного (нижнего) *F* и съемного (верхнего) *R*.

Снятие, установка, замена диска *R* выполняется только при включенном НМД.

Для *подготовки НМД* к работе необходимо:

проверить правильность положения переключателя СТАРТ (выключено, т. е. утоплен левый край переключателя);

включить НМД, т. е. утопить правый край переключателя сети; загорается соответствующая лампочка и появляется индикация ЗАГРУЗКА; далее можно установить или заменить диск *R* или продолжить включение;

включить двигатель переключателем СТАРТ (утопить правый край переключателя); при этом гаснет индикация ЗАГРУЗКА, появляется индикация ЗАЩИТА; примерно через минуту после включения СТАРТ раздается щелчок и появляется индикация ГОТОВО. Если же появилась индикация НЕИСПРАВНО, следует нажать на клавишу НЕИСПРАВНО, чтобы установить готовность НМД к работе. Если после нажатия индикация НЕИСПРАВНО сохранилась, нужно выключить НМД (см. ниже) и снова включить;

снять защиту записи нажатием клавиши ЗАЩИТА, если на диске будет выполняться запись; защита снимается одновременно на дисках *F* и *R*, а лампочка ЗАЩИТА гаснет.

Защиту на диске *F* можно установить командой

```
PRINT/1C HEX (1B0000000001001B1300); CR/LF
```

и на диске *R*—командой

```
PRINT/1C, HEX (1B0001000001001B1300); CR/LF
```

Замена диска выполняется при наличии индикации ЗАГРУЗКА; устанавливаемой либо при включении НМД, либо после выключения двигателя переключателем СТАРТ. Для замены необходимо выполнить последовательно действия:

поднять прозрачный козырек над сменным диском;

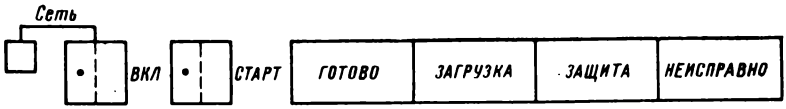


Рис. 5. Клавиши управления и индикация накопителя на магнитной ленте

поднять с усилием вверх по оси и затем отвести вправо до упора рычаг, фиксирующий крышку на диске;

снять съемную крышку с диска, перевернуть ее для дальнейшего использования в качестве футляра для снятого диска;

сдвинуть защелку контрастного цвета на ручке съемного диска и, удерживая защелку, поднять ручку;

поднять диск за ручку, прикладывая некоторое усилие, и опустить в перевернутую крышку, ручку на диске опустить до щелчка; в таком состоянии диск можно переносить за ручку и хранить;

установить на место снятого другой диск, вынув его предварительно из крышки футляра; для этого на опущенной ручке сдвинуть защелку, поднять диск, установить его в НМД по направляющим прорезям в диске и выступам на устройстве; далее опустить ручку диска, закрыть диск крышкой, зафиксировать крышку рычагом, опустить козырек.

Для *выключения НМД* необходимо:

выключить двигатель, утопив левый край переключателя СТАРТ; после полной остановки двигателя появится индикация ЗАГРУЗКА;

выключить питание НМД переключателем сети, утопив левый край переключателя.

4. ПОДГОТОВКА ПЕЧАТАЮЩИХ УСТРОЙСТВ К РАБОТЕ

«Роботрон 1154». Это печатающее устройство (ПУ) предназначено для вывода на печать алфавитно-цифровой и символьной информации.

Для *подготовки ПУ к работе* необходимо (рис. 6):

включить устройство нажатием переключателя NETZ на задней панели корпуса; при этом загорается лампочка *10* индикации питания (зеленого цвета);

заправить и закрепить бумагу;

перевести ПУ в рабочее положение. Красящую ленту менять по мере надобности.

Откидной рычаг *5* вызывает перемещение вала в горизонтальной плоскости. Шина *3* отрыва бумаги используется также для прижима бумаги к валу *4* сверху и для подсчета числа символов в строке как мерная линейка символов.

Торцовая кнопка *8* управляет вращением вала. Кнопка утоплена— вал можно поворачивать вручную, команды поворота вала от машины не выполняются. Кнопка отжата — поворот только по командам из машины.

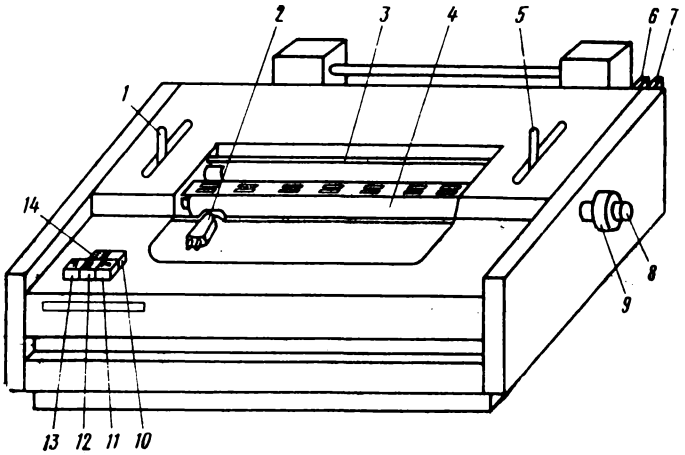


Рис. 6. Печатающее устройство «Роботрон 1154»:

1 — рычаг освобождения бумаги; *2* — печатающая головка; *3* — шина отрыва бумаги; *4* — бумагоопорный вал; *5* — откидной рычаг; *6* — кнопка блокировки клавиши подачи бумаги; *7* — переключатель NETZ; *8* — торцовая кнопка вала; *9* — внешняя часть (ручка) бумагоопорного вала; *10* — лампочка индикации питания; *11* — клавиша СИНХРОНИЗАЦИЯ; *12* — клавиша ВИДИМОСТЬ; *13* — клавиша подачи бумаги, совмещенная с лампочкой индикации; *14* — клавиша установки интервала

Клавиша *12* ВИДИМОСТЬ (практически не используется) при нажатии передвигает печатающую головку *2* на четыре деления (знака) вправо, и последний отпечатанный знак оказывается видимым. При отпускании клавиши головка возвращается в прежнее положение. Нельзя нажимать ее вместе с клавишей СИНХРОНИЗАЦИЯ.

Индикация НЕ ГОТОВО появляется, если откидной рычаг находится в положении «от себя», или печатающая головка не занимает крайнего левого положения, или неплотно прикрыта крышка корпуса устройства и т. д.

Рабочему состоянию устройства соответствуют следующие положения его элементов:

рычаг освобождения бумаги — «от себя»;

шина отрыва бумаги опущена;

откидной рычаг — «на себя»;

торцовая кнопка отжата;

печатающая головка в крайнем левом положении.

Обязательно наличие индикации включения сети и отсутствие индикации НЕ ГОТОВО, возможна индикация разблокировки клавиши подачи бумаги.

Для *заправки бумаги* необходимо:

перевести устройство в нерабочее положение, т. е. освободить бумагоопорный вал;

закрепить рулон на подающем валу;

заправить бумагу под бумагоопорный вал «на себя»;

выровнять бумагу по левому краю и закрепить на приемном валу;

перевести устройство в рабочее положение.

Подача бумаги «вручную» выполняется с помощью клавиши подачи бумаги при наличии в ней соответствующей индикации. Если вал прижат к печатающей головке, то одно нажатие на клавишу вызывает подачу бумаги на один интервал. Если вал отжат, то подача бумаги продолжается столько времени, сколько находится клавиша в утопленном положении, и прекращается при отпускании клавиши.

Подачу бумаги можно выполнить также поворотом вала за ручку при утопленной торцовой кнопке (использовать в крайнем случае, не забывая возвратить торцовую кнопку в исходное положение).

Выключается устройство клавишей NETZ.

5. ПОДГОТОВКА НАКОПИТЕЛЯ НА МАГНИТНОЙ ЛЕНТЕ К РАБОТЕ

НМЛ «Искра 005-61» на задней панели корпуса имеет тумблер вентилятора, на передней панели — лампочку индикации и переключатель сети, клавиши ЗАГРУЗКА, ДИСТАНЦ, ЗАЩИТА, ПЕРЕМОТКА, ВПЕРЕД, НАЗАД, СБРОС, совмещенные с лампочками индикации (рис. 7).

Устройство работает с ПИД в автоматическом рабочем режиме, что соответствует индикации ДИСТАНЦ.

Установку, загрузку, разгрузку, подмотку и перемотку ленты оператор выполняет с помощью клавиш НМЛ ВПЕРЕД, НАЗАД, ПЕРЕМОТКА в автономном режиме работы устройства НМЛ, что соответствует отсутствию индикации ДИСТАНЦ.

Для *подготовки НМЛ к работе* необходимо:

включить вентилятор переключателем на задней панели НМЛ справа внизу;

включить НМЛ переключателем сети; при этом устанавливается автономный режим, появляется индикация включения сети;

установить магнитную ленту; для чего открыть прозрачную крышку НМЛ, разблокировать замок на левой ступице НМЛ, нажав на конец рычага, помеченный кружочком контрастного цвета, установить катушку с лентой на вал и закрыть замок, отжав рычаг;

заправить свободный конец магнитной ленты (МЛ) согласно схеме и закрепить его (прижимом последующих витков) на приемном валу;

нажать на клавишу ЗАГРУЗКА; при этом лента начнет двигаться вперед, т. е. наматываться на приемную катушку; при фиксации фотодатчиком маркера физического начала движение прекращается, лента останавливается и загораются лампочки индикации ЗАГРУЗКА и

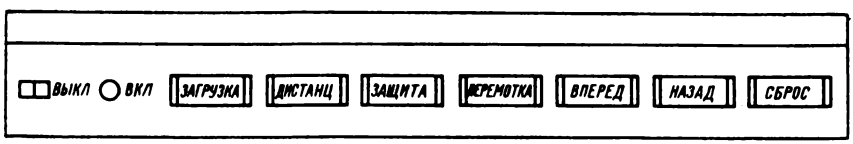


Рис. 7. Клавиши управления и индикация накопителя на магнитной ленте

ДИСТАНЦ, что означает готовность НМЛ к работе в автоматическом режиме.

Для *установки автономного режима*, а также для прерывания выполняемой подмотки (ВПЕРЕД, НАЗАД) и ускоренной перемотки (ПЕРЕМОТКА) используется клавиша СБРОС. Движущаяся назад лента останавливается автоматически при достижении маркера физического начала МЛ, а вперед — маркера физического конца ленты.

Для *установки автоматического режима* необходимо нажать на клавишу ДИСТАНЦ, когда лента не движется.

Для *разгрузки и снятия ленты* следует установить автономный режим работы устройства нажатием на клавишу СБРОС; затем нажать на клавишу ПЕРЕМОТКА и после остановки у маркера физического начала ленты при появлении индикации ЗАГРУЗКА еще раз нажать на клавишу ПЕРЕМОТКА. Ленту снять, выключить НМЛ и вентилятор.

6. ПОДГОТОВКА ГРАФОПОСТРОИТЕЛЯ Н-306 К РАБОТЕ

Для *подготовки графопостроителя к работе* необходимо:

проверить положение кнопок включения каналов «*X*» и «*Y*» (они должны быть отжаты);

установить лист диаграммной бумаги, совместив риски листа с рисками на столе прибора;

установить переключатели фиксированного смещения (СМЕЩ) в положение 0 по обоим каналам;

выставить среднее положение обоих каналов (—0— по каналу "X" и —0— по каналу «Y») ручкой установки начального положения;

заправить чернилами и установить на графопостроитель пишущее устройство (перо);

включить графопостроитель нажатием кнопки СЕТЬ на лицевой панели; при этом загорается лампочка рядом с кнопкой СЕТЬ;

установить масштаб регистрации, равный 0,25 В/см, нажав соответствующие кнопки на переключателях «*U*/см» сменных блоков обоих каналов;

закрепить диаграмму нажатием кнопки ДИАГР на лицевой панели прибора;

установить начальное положение регистрирующего устройства в левом нижнем углу листа диаграммной бумаги с помощью ручек —0— по каналу «*X*» и —0— по каналу «Y»;

включить каналы кнопками ВКЛ X и ВКЛ Y на лицевой панели прибора; через 30 мин после включения графопостроитель готов к работе.

По окончании работы графопостроитель необходимо выключить в следующем порядке:

отключить каналы, отжав кнопки ВКЛ X и ВКЛ Y;

освободить лист диаграммной бумаги, отжав кнопку ДИАГР;

выключить графопостроитель, отжав кнопку СЕТЬ.

7. ДИАГНОСТИКА ОСНОВНЫХ УСТРОЙСТВ МАШИНЫ ТЕСТАМИ В РЕЖИМЕ «ЗАГРУЗЧИК»

Режим ЗАГРУЗЧИК устанавливается при включении ПИД и после нажатия на кнопку SR (SYSTEM RESET) на передней панели ПИД.

Назначение кнопки SR. Кнопка SR предназначена для прерывания работы машины. Она используется в основном в режиме ЗАГРУЗЧИК, а в эксплуатационном режиме — только в экстренных случаях, например, когда заблокировалась клавиатура, сбилось ВМО и необходим специальный переход к режиму ЗАГРУЗЧИК.

Если кнопка SR нажата при выполнении диагностических тестов, машина переходит в исходное состояние (как при включении) и на экране индицируется ЗАГРУЗЧИК и ниже двоеточие и курсор.

Если кнопка SR нажата при выполнении каких-либо действий в режиме ЗАГРУЗЧИК (кроме диагностики), то происходит начальная установка УВВ, отмена отладочного режима, подготовка системной зоны оперативной памяти загрузчика. Содержимое УОП не изменяется. На экране индицируются RESET и ниже двоеточие и курсор.

Если кнопка SR нажата в эксплуатационном режиме, т. е. машина работает под управлением ВМО, то в дополнение к действиям предыдущего случая производится перегрузка системной зоны БЕЙСИК в УОП. На экране индицируется RESET и ниже двоеточие и курсор.

Выполнение диагностических тестов. Выполнение тестов начинается после набора на УК команды

RUN T CR/LF

Подчеркнутые одной чертой символы набираются с помощью одной клавиши, т. е. для выполнения данной команды необходимо нажать три клавиши: RUN, латинская буква T, CR/LF.

При использовании группы тестов диагностики процессора тесты, выполненные правильно, при индикации сопровождаются словом ПРОШЕЛ. Далее индицируются слова КОНФИГУРАЦИЯ МАШИНЫ и перечень ФАУ БИФ (см. табл. 3), входящих в состав машины, и ниже двоеточие и курсор.

При нажатии на клавишу CR/LF на экране индицируется список следующих тестов с номерами от 0 до 8 и с номером 8 слово ВЫХОД, а ниже — символы «вопрос» и «курсор» (см. с. 10), что означает ожидание ввода номера теста, который нужно выполнить или цифры 8 для естественного окончания диагностики. Набор цифры сопровождается нажатием клавиши CR/LF.

Например, для задания теста БОСИ на УК следует набрать 1 CR/LF, а для выхода из диагностики — 8 CR/LF. Чтобы тест выполнялся циклически, т. е. повторялся до прерывания SR, нужно набрать после номера теста букву R. Например, 1R CR/LF.

Каждый из перечисленных тестов для проверки УВВ исполняется в соответствии с инструкцией и описанием теста в документе «Входной язык ЗАГРУЗЧИКА».

8. ЗАГРУЗКА ВНУТРЕННЕГО МАТЕМАТИЧЕСКОГО ОБЕСПЕЧЕНИЯ И УСТАНОВКА ЭКСПЛУАТАЦИОННОГО РЕЖИМА

Загрузка означает считывание ВМО с диска и ввод его в УОП и выполняется в режиме ЗАГРУЗЧИК. Если нет уверенности в том, что машина находится в этом режиме, его следует установить нажатием на кнопку SR и выполнить следующие действия:

1) в НГМД (*F* или *R*) установить диск с ВМО (см. с. 20). Этот диск поставляется с обозначением#0,1, 2, 3 , где цифры — номера четырех копий ВМО;

2) набрать на клавиатуре команду загрузки одной из четырех копий ВМО:

LOAD ⟨к⟩ # ⟨н⟩ CR/LF,

где <к> — тип диска (фактически набирается *F* или *R* в зависимости от того, в какой дисковод установлен диск); <н> — номер копии (набирается цифра 0 или 1, или 2, или 3).

Например, диск установлен в нижний дисковод ЕС 5074 (т. е. *F*), нужно загрузить копию 0 (в отличие от буквы О цифра ноль перечеркивается). Для выполнения этой операции следует набрать на клавиатуре команду

LOAD F # 0 CR/LF

Подчеркнутые сплошной чертой символы набираются нажатием одной клавиши.

После окончания загрузки на экране индицируется сообщение

BASIC 01 ⟨дата⟩

Здесь 01 — номер версии ВМО.

Если индикации нет, загрузку следует повторить, предварительно нажав кнопку SR;

3) задать эксплуатационный режим, набрав на клавиатуре команду

RUN 1 CR/LF

На экране появится индикация готовности работы с системой БЕЙСИК:

READY

—

В исполнениях 5 и 6 ПЭКВМ «Искра 226» можно загрузить ВМО с КНМЛ. Для этого необходимо:

установить КМЛ и перемотать ее в начало;

выполнить набранную на клавиатуре команду

LOAD F/08, # 1 CR/LF и далее RUN 1 CR/LF

На экране появится индикация готовности

```
READY
:_
```

Если индикация отлична от указанной, то загрузку ВМО следует повторить. Для этого нажать на кнопку SR и продолжить либо с команды RUN 1 CR/LF, либо с команды считывания копии ВМО.

Глава III
УПРАВЛЕНИЕ РАБОТОЙ МАШИНЫ

1. ОБЩИЕ СВЕДЕНИЯ

Все изложенное в этой главе относится к эксплуатационному режиму при загруженном ВМО, точнее к работе машины под управлением БЕЙСИК-системы.

БЕЙСИК-система обеспечивает реализацию входного языка БЕЙСИК, а ВМО — более общее понятие; оно может быть системой математического обеспечения другого алгоритмического языка, как в любой ЭВМ с загружаемым математическим обеспечением. В комплекте поставки машины «Искра 226» ВМО — это БЕЙСИК-система.

ПЭКВМ «Искра 226» может работать в режиме непосредственного счета (НС) и в режиме счета по программе, но всегда исполняются операторы входного языка, т. е. программа, поэтому машину и называют программоуправляемой. Указанные режимы имеют следующие отличия:

программа в режиме НС ограничена по числу составляющих ее символов — не более 240 символов, т. е. в пределах одной логической строки экрана при наборе (см. с. 10);

программа в режиме НС вводится только с клавиатуры в виде одной строки программы без номера, поэтому далее под понятием «строка НС» следует понимать «программа, выполняемая в непосредственном счете»;

строка НС исполняется сразу же после завершения набора и не записывается в зону программы ОП;

программа в режиме счета по программе (в дальнейшем — просто программа) ограничена емкостью ОП и числом строк 9999);

программа вводится с клавиатуры и с технических носителей;

введенная программа хранится в ОП в зоне программы и может быть исполнена лишь при задании режима счета.

Не все операторы входного языка исполняются в непосредственном счете.

Режим непосредственного счета принято считать режимом калькулятора, когда выполняются какие-либо вычисления. Чаще режим непосредственного счета используется для управления работой машины в режиме счета по программе.

Итак, управление работой машины вручную с клавиатуры осуществляется вводом управляющих команд.

Управляющая команда — это строка НС, т. е. один или несколько операторов языка, выполняемых в непосредственном счете, или команда, поданная нажатием клавиши-команды HALT/STEP, CONTINUE или RESET и т. д. (клавиши красного цвета).

2. ОЧИСТКА ОПЕРАТИВНОЙ ПАМЯТИ

Оперативная память очищается от введенной в нее информации полностью или частично по зонам оператором CLEAR:

CLEAR CR/LF — команда очистки памяти, т. е. она стирает и текст программы, и данные, и информацию служебной зоны (см. с. 7); набирается нажатием двух клавиш последовательно.

CLEAR V CR/LF — стирает только все данные, текст программы сохраняется;

CLEAR N CR/LF — стирает необщие данные, общие, т. е. перечисленные ранее в операторе COM (см. с. 72), и текст программы сохраняются;

CLEAR P CR/LF — стирает текст программы, все данные сохраняются.

Если после параметра P указан один или два номера строки, то стирается часть программы, т. е. строки программы в заданном диапазоне номеров;

CLEAR P 20, 70 CR/LF стирает с 20-й до 70-й строки, т. е. со строки с первым номером до строки со вторым номером;

CLEAR P 20 CR/LF стирает с 20-й строки до конца программы, т. е. наличие в команде первого номера строки означает стирание части программы с этого номера строки до конца;

CLEAR P, 70 CR/LF стирает с начала программы до 70-й строки, т. е. наличие в команде только второго номера строки (об этом свидетельствует запятая перед номером) означает стирание части программы от начала до строки с указанным номером.

CLEAR P: CLEAR N CR/LF стирает программу и необщие данные. Эта команда состоит из двух операторов, которые могут быть выполнены и как отдельные команды.

3. ВВОД ПРОГРАММЫ

Ввод программы с клавиатуры. Строка НС, как и строка программы, может состоять из одного или нескольких операторов, разделенных символом «двоеточие» (:).

Строку можно набирать, когда в последней строке экрана индикация двоеточие и курсор (:_), т. е. когда машина находится в режиме ожидания ввода команды. Текст строки набирается нажатием информационных клавиш. Введенное значение индицируется на экране. Имя оператора и его параметры можно набирать либо нажатием клавиши, содержащей имя или параметр полностью, либо последовательным нажатием клавиш с буквами латинского алфавита, например для ввода опе-

ратора PRINT можно нажать одну клавишу PRINT в четвертой зоне, либо нажать последовательно клавиши с латинскими буквами P R I N T (без дополнительных символов между ними) в первой зоне. Для удобства контроля за набором введенное имя оператора автоматически сопровождается пробелом на экране. Пробелы, как и буквы русского алфавита, используются лишь при наборе текста, заключенного в кавычки и апострофы (исключение составляют операторы REM и %):

```
: PRINT „ОТЧЕТ _ЗА_ ''; A%; „КВАРТАЛ"
```

Знак _ соответствует символу «пробел», который нужно набирать обязательно.

Нельзя пользоваться клавишами с буквами русского алфавита при наборе латинских букв, имеющих одинаковое начертание с русскими (A, B, C, K, ...)

Допускается нажатие на клавишу «Пробел» между параметрами, так как символы при контроле вводимой информации автоматически игнорируются.

Если в строке есть оператор BACK SPACE, то его набор выполняется одной или несколькими клавишами первой зоны, но не клавишей с тем же названием, расположенной во второй зоне.

Завершается набор строки нажатием клавиши CR/LF.

Строка программы отличается от строки НС наличием номера, располагаемым в начале строки. Номер набирается цифровыми клавишами первой или третьей зоны или специальной клавишей STMT NUMBER, при этом нумерация строк имеет шаг 10.

Если при наборе допущена синтаксическая ошибка и машина обнаружила ее, то на экране появляется индикация ERROR 06 (06 — номер ошибки), но строка не игнорируется и так же, как правильная, заносится в оперативную память: строка НС в свою зону, а строка программы — в зону программы. Ошибочная строка НС не исполняется, но ее можно отредактировать.

После нажатия CR/LF в строке НС может появиться индикация ошибки с номером, отличным от 06, что также означает невозможность выполнения правильно набранной строки по каким-либо другим причинам: не готово устройство, не снята защита записи с диска и т. д.

Ввод программы с магнитного диска. Ввод программы с технического носителя в зону программы для исполнения, в отличие от ввода данных, называют *загрузкой* программы. Далее в тексте могут встретиться оба термина.

Загрузка возможна с жесткого и гибкого магнитных дисков, с кассетной магнитной ленты (см. с. 135, 139).

Рассмотрим лишь команду загрузки программы с гибкого диска. Эта команда используется для загрузки тестовых и демонстрационных программ, поставляемых заводом с машиной, а также для загрузки любой программы, записанной в режиме каталога (см. с. 117).

```
LOAD DC ⟨тип диска⟩ „⟨имя⟩" CR/LF
```

<тип диска> — F или R в зависимости от того, где установлен диск с программой (см. с. 15, 20); <имя> — имя программы, с которым программа записана на диске.

Память перед загрузкой должна быть очищена.

Например, чтобы загрузить программу с именем ТЕСТ_1.6, необходимо диск с библиотекой тест-программ установить в НГМД F (см. с. 20), выполнить последовательно команды очистки памяти и загрузки программы:

```
CLEAR CR/LF
LOAD DC F "ТЕСТ_1.6 "CR/LF
```

Имя программы в примере набирается буквами русского алфавита.

Можно эти команды объединить в одну:

```
CLEAR : LOAD DC F "ТЕСТ_1.6" CR/LF
```

4. ВЫЗОВ ПРОГРАММЫ ИЗ ОПЕРАТИВНОЙ ПАМЯТИ НА ЭКРАН И РАСПЕЧАТКА ТЕКСТА ПРОГРАММЫ

Чтобы проверить наличие текста программы в оперативной памяти или просмотреть этот текст, его можно вызвать на экран БОСГИ полностью, если текст программы умещается на экране, или по страницам.

Операторы вызова:

LIST CR/LF — вызов всей программы; если вызванная программа не уместилась на экране, то индицируются последние строки в пределах объема экрана;

LIST S CR/LF — вызов первой страницы текста программы;

CR/LF — вызов каждой последующей страницы текста программы;

LIST/0C CR/LF — распечатка текста программы.

Оператор LIST имеет более широкое применение; он рассмотрен подробнее на с. 111—113.

5. РЕДАКТИРОВАНИЕ ТЕКСТА ПРОГРАММЫ

Редактирование текущей строки программы. Текущая строка — это та строка программы на экране, в которой находится курсор.

Текущей может быть и строка программы, и строка НС, т. е. та строка, которая набирается или редактируется.

Если текущая строка — строка набора и в начале строки имеется символ двоеточие (:), то редактировать можно с помощью следующих клавиш:

BACK SPACE — стирание символа, предшествующего курсору;

LINE ERASE — стирание всей строки.

Пример 1.

```
:PRINT "НАИМЕООВАНИЕ_
```

Нажать на клавишу BACK SPACE 7 раз, в строке останутся символы

```
:PRINT "НАИМЕ —
```

Далее продолжить набор правильно.

Чтобы сделать строку набора *редактируемой*, необходимо нажать на клавишу EDIT.

Если символ «двоеточие» (:) в начале строки заменить символом «звездочка» (*), строку можно редактировать, используя клавиши шестой и восьмой зон клавиатуры.

Возможны следующие варианты редактирования.

Заменить символ (несколько символов подряд):

подвести курсор клавишами восьмой зоны к заменяемому символу (первому из заменяемых);

нажать на клавишу с символом замены.

Пример 2.

:PRINT "НАИМЕООВАНИЕ —

Нажать последовательно клавиши EDIT, ← — — и дважды клавишу ←. Строка примет вид:

* PRINT "НАИМЕООВАНИЕ

Далее нажать клавиши Н, — — →, → и продолжить набор.

Вставить символ (несколько символов):

подвести курсор к месту вставки клавишами восьмой зоны;

нажать на клавишу INSERT столько раз, сколько будет вставляться символов;

набрать символ (символы) вставки.

При нажатии INSERT курсор остается на месте, а над ним появляется пробел.

Пример 3.

:PRINT "НАИМЕВАНИЕ_

Нажать клавиши EDIT, ← — — и дважды INSERT.
Строка примет вид:

*PRINT "НАИМЕ _ _ ВАНИЕ

Далее нажать клавиши, Н, О — — → и продолжить набор.

Удалить символ (несколько символов подряд):

подвести курсор к удаляемому символу (первому из удаляемых символов) клавишами восьмой зоны;

нажать на клавишу DELETE столько раз, сколько символов удаляется.

Пример 4.

:PRINT "НАИМЕНОВАННИЕ —

Нажать на клавишу EDIT и трижды на клавишу ←. Строка примет вид:

* PRINT "НАИМЕНОВАННИЕ

Далее нажать на клавишу DELETE, дважды на клавишу → и продолжить набор.

Если удаляемые символы последние в строке, то для их удаления необходимо:

подвести курсор к первому удаляемому символу;

нажать на клавишу ERASE.

Вызов строки программы из оперативной памяти для редактирования. Набираемая строка программы после нажатия на клавишу CR/LF перестает быть текущей, она записывается в зону программы оперативной памяти, а ее изображение становится недоступным для перемещения курсора. Чтобы отредактировать любую строку программы, находящуюся в памяти машины, ее нужно сделать текущей, т. е. вызвать на экран для редактирования, внести исправления в режиме EDIT (см. с. 33) и возвратить в оперативную память нажатием клавиши CR/LF.

Вызов строки выполняется по номеру, для чего необходимо:

нажать на клавишу EDIT;

набрать номер строки;

нажать на клавишу RECALL.

После окончания редактирования курсор можно оставить в любой позиции редактируемой строки, не возвращая его в конец.

Если в вызванной для редакции строке заменяется номер строки, то строка запишется в оперативную память с новым номером и сохранится там со старым номером. Если строка со старым номером не нужна, ее следует удалить.

При наборе программы с большим числом строк и большим числом редактируемых строк может произойти переполнение оперативной памяти и дальнейший набор окажется невозможным. В этом случае необходимо вывести программу на диск (см. с. 136), очистить память, загрузить программу с диска и продолжить набор.

Строка НС после нажатия на клавишу CR/LF исполняется и записывается в зону строки НС оперативной памяти. Ее так же, как и строку программы, можно вызвать для редактирования или для повторного исполнения без редактирования. Так как в памяти строка НС только одна (та, которая была набрана последней), то она и будет вызываться. Для вызова необходимо нажать на клавиши: EDIT и RECALL.

По окончании редактирования для исполнения строки непосредственного счета следует нажать на клавишу CR/LF.

Редактирование строк программы. Возможны следующие варианты.

Заменить строку программы — это значит набрать новый текст строки с номером заменяемой строки.

Удалить строку программы — заменить ее пустой строкой, т. е. набрать номер удаляемой строки и нажать на клавишу CR/LF.

Удалить одну или несколько строк можно также операторами CLEAR (см. с. 62).

Вставить строку — набрать вставляемую строку с номером из числа резервных номеров, т. е. чтобы вставить строку между строками 70 и 80, ее нужно набрать с одним из номеров: 71, 72, 73, 74, 75, 76, 77, 78, 79. Номер набирается цифровыми клавишами.

Если резерва нет, т. е. строки пронумерованы с шагом 1 или число вставляемых строк больше числа резервных номеров, то необходимо выполнить перенумерацию строк программы:

RENUMBER CR/LF

Все строки программы перенумеруются с шагом 10, т. е. появятся резервные номера.

Программу после перенумерации следует просмотреть (пролистать) для уточнения места вставки (см. с. 32) и затем набрать строку с резервным номером.

Можно выполнить перенумерацию лишь части программы, причем шаг может быть отличен от 10 (см. с. 63)

6. ЗАПУСК ПРОГРАММЫ НА СЧЕТ

Чтобы исполнить программу, введенную в память машины, необходимо нажать клавиши:

RUN CR/LF

При этом произойдет начальная подготовка к счету по программе (см. с. 64) и затем программа начнет исполняться с первого оператора первой строки программы. Первая строка может иметь любой номер, например 10, 75 и т. д.

Не следует выполнять запуск программы на счет с указанием номера первой строки:

RUN 10 CR/LF

так как отсутствие начальной подготовки к счету может привести к сбойной ситуации.

7. ОСТАНОВЫ СЧЕТА ПО ПРОГРАММЕ

Прекращение счета по программе означает переход машины в режим ожидания ввода управляющей команды (индикация:_), т. е. в режим непосредственного счета. Прекращение счета называют *остановом*. В зависимости от причин, вызывающих остановы, различают несколько типов остановов.

Естественный останов — окончание счета по программе происходит автоматически после выполнения последнего оператора программы или оператора окончания счета END.

Программный останов, т. е. останов, предусмотренный заранее в программе, происходит при выполнении оператора STOP. На экране индицируется номер строки, слово STOP и возможный текст.

Такой останов дает возможность выполнить какие-либо действия, соответствующие технологии счета: заправить бумагу, заменить диск и затем продолжить счет по программе. Используется этот оператор так-

же при отладке программы. Он позволяет пользователю просмотреть на экране промежуточные результаты счета. Число операторов STOP в программе не ограничено. Продолжить выполнение программы можно нажатием на клавишу CONTINUE. Выполнение программы начинается с оператора, следующего за оператором STOP, если выполнение возможно, т. е. если нет сбойной ситуации.

Прерывание счета пользователем с клавиатуры выполняется нажатием на клавишу HALT/STEP. Остановка счета происходит после завершения исполняемого в данный момент оператора программы. Используется прерывание для тех же целей, что и оператор STOP. Продолжить счет можно нажатием на клавишу CONTINUE.

Аварийный останов выполняется пользователем с клавиатуры нажатием на клавишу RESET или (в исключительных случаях) нажатием на клавишу SR на передней панели ПИД. Прерывание счета происходит сразу же после нажатия, до завершения исполняемого оператора, поэтому продолжить счет по программе после аварийного останова невозможно. Используется останов, например, если в печатающем устройстве смялась бумага или продолжать счет нецелесообразно и его нужно прекратить.

После нажатия на клавишу RESET программа и данные сохраняются в ОП, и можно повторить выполнение программы с начала по команде RUN CR/LF или по команде RUN <номер строки> CR/LF с какой-либо строки программы, определяющей начало технологического этапа счета, с которого можно повторить счет. Такая возможность должна быть предусмотрена заранее при программировании.

После нажатия на клавишу SR устанавливается режим ЗАГРУЗЧИК (см. с. 27). В случае сохранения в памяти машины программы пользователя, внутреннего математического обеспечения и служебной информации программу можно выполнить с начала. По команде загрузчика RUN 2 CR/LF устанавливается эксплуатационный режим, затем по команде RUN CR/LF исполняется программа. Если программа пользователя не сохранилась, необходимо восстановить эксплуатационный режим командой RUN 1 CR/LF, загрузить программу в ОП и выполнить ее по команде RUN CR/LF.

Останов счета при возникновении ошибки происходит в том случае, если дальнейшее выполнение программы невозможно. На экране индицируется сообщение об ошибке в виде номера строки, в которой произошла ошибка, и номер ошибки (табл. 4). К таким ошибкам относятся недопустимое использование операторов и их параметров, ошибки при выполнении матричных и дисковых операторов, ошибки ввода-вывода при сбойной работе устройств ввода-вывода, например, ERR 03— деление на нуль, ERR 80 01 — авария устройства.

Если ошибка устранима, то после ее устранения можно продолжить счет по программе нажатием на клавишу CONTINUE. Например, индикация ERR 80 03 означает «устройсто ввода-вывода не готово к обмену». Для продолжения счета можно перевести устройство в состояние готовности и нажать на клавишу CONTINUE.

4. Перечень ошибок

Номер ошибки	Характер ошибки
SYSTEM ERROR 01	Сбой ОЗУ
SYSTEM ERROR 02	Сбой ПЗУ
ERR 01	Переполнение текста
ERR 02	Переполнение таблиц
ERR 03	Математическая ошибка
ERR 04	Неопределенный массив или переменные при вводе строки HC
ERR 05	Резерв
ERR 06	Ошибка трансляции строки при вводе с клавиатуры
ERR 07	Ошибка трансляции программной строки при вводе с МЛ (диска)
ERR 08	Не определена функция FN
ERR 09	Несоответствие формальных и фактических параметров DEFFN′ и GOSUB′
ERR 10	Синтаксическая ошибка при вводе программы из переменной по оператору LOAD
ERR 11	Несуществующий номер строки
ERR 12	Неправильный оператор
ERR 13	Резерв
ERR 14	Резерв
ERR 15	Неправильный номер строки или диапазон номеров строк
ERR 16	Резерв
ERR 17	Резерв
ERR 18	Недопустимая величина (числа, размерности, массива, индекса)
ERR 19	Резерв
ERR 20	Недопустимый формат числа
ERR 21	Резерв
ERR 22	Неопределенный массив или переменная, встретившиеся при выполнении программы
ERR 23	Программы нет в ОП
ERR 24	Неупорядоченный список данных
ERR 25	Недопустимая пара операторов GOSUB/RETURN или FOR/NEXT
ERR 26	Резерв
ERR 27	Данные исчерпаны. Обращение к данным за допустимыми пределами
ERR 28	Резерв
ERR 29	Недопустимый формат данных в операторе INPUT
ERR 30	Резерв
ERR 31	Недопустимый номер строки
ERR 32	Длина загружаемой по оператору SAVE программы больше длины переменной
ERR 33	Резерв
ERR 34	Резерв
ERR 35	Резерв
ERR 36	Недопустимый формат данных в операторах CONVERT, ¤ PACK, ¤ UNPACK, IMAGE
ERR 37	Отсутствует оператор %
ERR 38	Резерв
ERR 39	Резерв
ERR 40	Задана неверная последовательность переменных в операторе LIST V

Продолжение табл. 4

Номер ошибки	Характер ошибки
ERR 41	Недопустимый аргумент функции STR
ERR 42	Резерв
ERR 43	Недопустимое присвоение
ERR 44	Защищенная программа
ERR 45	Слишком длинный оператор
ERR 46	Новый начальный номер строки слишком мал
ERR 47	Недопустимое значение адреса внешнего устройства
ERR 48	Нет подпрограммы DEFFN′ с указанным номером
ERR 49	Неквадратная матрица
ERR 50	Матричные операнды несовместимы
ERR 51	Недопустимый матричный операнд
ERR 52	Сингулярная матрица
ERR 53	Массив слишком мал
ERR 54	Недопустимая длина строки устройства
ERR 55	Ошибочный адрес сектора
ERR 56	Число превышает формат
ERR 57	Недопустимый адрес сектора
ERR 58	Необобщенная переменная
ERR 59	Резерв
ERR 60	Файл не открыт или не существует
ERR 61	Резерв
ERR 62	Резерв
ERR 63	Резерв
ERR 64	Резерв
ERR 65	Резерв
ERR 66	Ошибка по КС при считывании МЛ
ERR 67	Неправильный тип записи программа/данные
ERR 68	Буфер ввода-вывода мал
ERR 69	Переменная/массив не помещается в буфер ввода-вывода
ERR 70	В операторе PRINTUSING нет формата
ERR 71	Файл уже есть в каталоге
ERR 72	Файл вычеркнут
ERR 73	Файл отсутствует в каталоге
ERR 74	Недопустимый адрес сектора
ERR 75	Конец каталога
ERR 76	Файл заполнен
ERR 77	Временный файл попадает в область каталога
ERR 78	Указатель каталога заполнен
ERR 79	Резерв
ERR 80	Ошибка ввода-вывода:
01	авария УВВ (диск, лента ...)
02	нарушение последовательности процедур
03	УВВ не готово к обмену
04	резерв
05	защита от записи
06	маркер начала (конца) ленты
07	ошибка по КС контрольного чтения после записи

Номер ошибки	Характер ошибки
08	авария БИФ
09	резерв
0A	ошибка по КС при считывании
0B	ошибка процедуры (кодирование, формат, КС обмена ПИД и БИФ)
0C	сектор с указанным адресом не найден
0D	нарушение последовательности интерфейсных команд в процедуре
0E	недопустимый адрес сектора
0F	ошибочная длина массива

Глава IV
КОДИРОВАНИЕ ИНФОРМАЦИИ

1. СИСТЕМЫ СЧИСЛЕНИЯ

Усвоив с детства правила записи чисел и правила выполнения арифметических действий над ними, мы не задумываемся над тем, какой системой счисления пользуемся. Это десятичная система счисления с позиционным принципом записи, согласно которому один и тот же символ имеет различные значения в зависимости от места, которое он занимает.

Система счисления — это совокупность символов, используемых для записи числа, и правил записи чисел с помощью этих символов. Символы называются цифрами. Изображенное число — последовательность цифр.

В позиционной системе счисления позиции цифр числа нумеруются справа налево в целой части и отрицательными числами слева направо в дробной части. Например, десятичное число 292, 42 условно можно записать в виде $a_2a_1a_0$, $a_{-1}a_{-2}$, где a_i — цифра i-го разряда числа.

Число можно представить в виде суммы значений его цифр:

$$292{,}42 = 200 + 90 + 2 + 0{,}4 + 0{,}02 = 2 \times 10^2 + 9 \times 10^1 + \\ + 2 \times 10^0 + 4 \times 10^{-1} + 2 \times 10^{-2}.$$

В общем случае можно записать

$$a_n\, a_{n-1} \ldots a_1\, a_0,\, a_{-1} \ldots a_{-m} = a_n\, p^n + a_{n-1}\, p^{n-1} + \ldots + a_1\, p + \\ + a_0 + a_{-1}\, p^{-1} + \ldots + a_{-m}\, p^{-m},$$

где p — основание системы счисления, равное количеству различных цифр, используемых для данной p-ичной системы счисления. Это основное свойство позиционной системы счисления. В системе счисления с основанием p число p всегда изображается как 10 (табл. 5).

5. Сравнительные параметры систем счисления

Система счисления	Значение основания	Цифры системы счисления
Десятичная	$p=10$	0, 1, 2, 3, 4, 5, 6, 7, 8, 9
Восьмеричная	$p=8=10_8$	0, 1, 2, 3, 4, 5, 6, 7
Двоичная	$p=2=10_2$	0, 1
Шестнадцатеричная	$p=16=10_{16}$	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F

Число 14 в двоичной системе счисления изображается в виде 1110, в шестнадцатеричной — E, т. е. $14 = 1110_2 = E_{16}$. Чтобы сравнить значения чисел, записанных в разных системах счисления, их нужно перевести в одну систему счисления. Для перевода чисел из одной системы счисления в другую существуют специальные правила, однако для перевода целых чисел с малой разрядностью можно воспользоваться представлением числа в виде суммы значений его цифр в соответствующей системе счисления и табл. 6. Например,

$$1110_2 = 1 \times 10_2^3 + 1 \times 10_2^2 + 1 \times 10_2 + 0 \times 10_2^0 = 1 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 = 1 \times 8 + 1 \times 4 + 1 \times 2 + 0 = 8 + 4 + 2 = 14;$$

$$1F_{16} = 1 \times 10_{16}^1 + F_{16} \times 10_{16}^0 = 1 \times 16^1 + 15 \times 16^0 = 16 + 15 = 31.$$

Так как 16 является степенью числа 2, то перевод из шестнадцатеричной системы счисления в двоичную выполняется заменой каждой шестнадцатеричной цифры четырехразрядным двоичным числом в соответствии с табл. 6:

$$1F_{16} \quad 0001\ 1111_2 \quad 11111_2$$

6. Запись чисел в различных системах счислениях

Десятичная	Двоичная	Шестнадцатеричная	Десятичная	Двоичная	Шестнадцатеричная
1	1	1	17	1 0001	11
2	10	2	18	1 0010	12
3	11	3	19	1 0011	13
4	100	4	20	1 0100	14
5	101	5	21	1 0101	15
6	110	6	22	1 0110	16
7	111	7	23	1 0111	17
8	1000	8	24	1 1000	18
9	1001	9	25	1 1001	19
10	1010	A	26	1 1010	1A
11	1011	B	27	1 1011	1B
12	1100	C	28	1 1100	1C
13	1101	D	29	1 1101	1D
14	1110	E	30	1 1110	1E
15	1111	F	31	1 1111	1F
16	1 0000	10	32	10 0000	20

И наоборот, каждой четверке двоичных разрядов ставится в соответствие шестнадцатеричная цифра. Разбиение двоичного числа начинается от запятой в целой части влево, в дробной — вправо. Например, $1001110001_2 \rightarrow 10.0111.0001_2 \rightarrow 271_{16}$.

Один двоичный разряд (0 или 1) является минимальной единицей информации и называется *битом*, четыре двоичных разряда (0000—1111) — тетрадой, восемь двоичных разрядов (0000 0000 — 1111 1111) — байтом. Часто используется еще одна единица — К байт.

1024 байт = 1К байт.

2. КОДЫ ИНФОРМАЦИИ

Вся информация, обрабатываемая вычислительными машинами, представляется в виде числовых кодов. В разных машинах используется свой способ кодирования, но при этом во всех современных вычислительных машинах для кодирования информации применяется двоичная система счисления. В ПЭКВМ «Искра 226» двоичная система счисления используется только для внутреннего кодирования, т. е. пользователь не оперирует непосредственно двоичными числами при программировании, вводе программы и данных. Он записывает числа в привычной десятичной системе счисления, а текст (символьные данные) обычными символами, имеющимися на клавиатуре, и только иногда, например для записи и ввода отсутствующего на клавиатуре символа, использует его шестнадцатеричный код.

При вводе информации в машину осуществляется автоматическое кодирование, при этом каждая цифра числа заменяется двоичной тетрадой в соответствии с табл. 6, а каждый символ текста (буква,цифра, знак) — байтом. Значение байта равно двухразрядному шестнадцатеричному коду символа, который называется HEX-кодом символа (от слова HEXADECIMAL — шестнадцатеричный).

Совокупность HEX-кодов (табл. 7) используется также в машинах серии ЕС и серии СМ и носит название КОИ-8 (ГОСТ 19768—74).

Например, чтобы представить в закодированном виде название машины «Искра 226» нужно воспользоваться сначала таблицей кодов (см. табл. 7), а затем заменить каждую цифру полученного HEX-кода двоичной тетрадой (см. табл. 6). HEX-код получается из номера столбца (старший разряд кода) и номера строки (младший разряд кода). Так, букве И соответствует номер столбца Е и номер строки 9 (см. табл. 7), поэтому ее код будет Е9. Тогда «ИСКРА 226» закодируется следующим образом:

И — Е9 — 1110 1001, С — F3 — 1111 0011, К — ЕВ — 1110 1011,

Р — F2 — 1111 0010, А — Е1 — 1110 0001, _ — 20 — 0010 0000,

2—32—0011 0010, 2—32—0011 0010, 6—36—0011 0110,

т.е. «ИСКРА 226» имеет код 1110 1001 1111 0011 1110 1011 1111 0010 1110 0001 0010 0000 0011 0010 0011 0010 0011 0110

7. Шестнадцатеричные коды символов (HEX-коды)

Младшая цифра кода	Старшая шестнадцатеричная цифра кода															
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	ПУС	АР1	Пробел	0	@	P	`	p	ВЦФ	Д16	≡		ю	п	Ю	П
1	НЗ	СУ1	!	1	A	Q	a	q	НЗН	Д17			а	я	А	Я
2	НТ	СУ2	"	2	B	R	b	r	РП	УУК			б	р	Б	Р
3	КТ	СУ3	#	3	C	S	c	s	Д03	Д19			ц	с	Ц	С
4	КП	СТП	¤	4	D	T	d	t	БК	ВКП			д	т	Д	Т
5	КТМ	НЕТ	%	5	E	U	e	u	НС	ССУ			е	у	Е	У
6	ДА	СИН	&	6	F	V	f	v	НП	ВП			ф	ж	Ф	Ж
7	ЗВ	КБ	'	7	G	W	g	w	ОЖД	Д23			г	в	Г	В
8	ВШ	АН	(	8	H	X	h	x	Д08	Д24			х	ь	Х	Ь
9	ГТ	КН	)	9	I	Y	i	y	Д09	Д25			и	ы	И	Ы
A	ПС	ЗМ	*		J	Z	j	z	УР	Д26			й	з	Й	З
B	ВТ	АР2	+	;	K	[	k	{	СП2	СП3			к	ш	К	Ш
C	ПФ	РФ	,	<	L	\	l	|	Д12	ВЫП		л	э	Л	Э
D	ВК	РГ	—	=	M	]	m	}	Д13	ВСТ			м	щ	М	Щ
E	ВЫХ	РЗ		>	N	↑	n	~	НРВ	Д30			н	ч	Н	Ч
F	ВХ	РЭ		?	O	—	o	≡	СП1	Д31			0	ъ	О	ЗБ

Глава V

БЕЙСИК — ВХОДНОЙ ЯЗЫК МАШИНЫ

БЕЙСИК — алгоритмический язык высокого уровня, мощный, гибкий, удобный для программирования. Язык имеет операторную структуру. БЕЙСИК используется для общения с машиной, т. е. для записи и ввода (а также вывода) программы, которую машина должна выполнить, и данных, участвующих в обработке команд. Как любой язык, БЕЙСИК имеет свои синтаксические правила построения конструкций языка. Для описания синтаксиса удобно пользоваться стандартными лингвистическими обозначениями. Они не всегда позволяют точно и однозначно описать конструкцию языка и при первом прочтении кажутся лишними, ненужными, загромождающими текст и память читающего. Однако компактность записи конструкции, обеспечиваемая этими обозначениями, очень удобна, особенно при повторных прочтениях.

1. ОБОЗНАЧЕНИЯ СИНТАКСИЧЕСКОГО ОПИСАНИЯ

При синтаксическом описании используются следующие обозначения.

< > — обозначение элемента в конструкции. В программе вместо обозначения элемента должен быть записан сам элемент, его фактическое значение. Такое обозначение уже использовалось (см. с. 31) для обозначения имени программы и типа диска. В описании это обозначение используется также при определении элемента.

[ ] — обозначение необязательного элемента конструкции, т. е. этот элемент в конструкции может отсутствовать и его отсутствие не приводит к синтаксической ошибке.

: := — равно по определению, иначе «это есть».

| — «или». Используется для перечисления возможных значений только в определении элемента.

$\left\{\begin{matrix}<:>\\<:>\end{matrix}\right\}$ — обозначение списка элементов, один из которых должен быть в конструкции.

{ }... — многократное повторение элемента.

Эти обозначения можно использовать в различных сочетаниях. Одну и ту же конструкцию можно описать по-разному.

2. СИНТАКСИС ОСНОВНЫХ КОНСТРУКЦИЙ ЯЗЫКА

Структура программы. Программа состоит из одной или нескольких пронумерованных строк. Номер строки — целое число в диапазоне от 0 до 9999. Номера можно использовать не подряд, а с каким-либо шагом, для разных участков программы шаг может быть разным. Номера строк задают порядок выполнения программы. Естественная последовательность выполнения строк (по возрастанию их номеров) может быть нарушена программным переходом. Номер строки является меткой для

такого перехода. Вводить строки программы можно в любой последовательности, т. е. не обязательно по возрастанию номеров.

Строка программы в свою очередь состоит из одного или нескольких операторов языка, разделенных символом «двоеточие» (:). CR/LF обычно в программе не записывается, но ввод строки в память машины не произойдет до тех пор, пока клавиша CR/LF не будет нажата. Код клавиши воспринимается без сопровождающей индикации на экране.

Оператор состоит из имени оператора и тела оператора в виде совокупности каких-либо параметров, элементов и т. д. Исключение составляют операторы END и RETURN, они имеют только имя.

Формально структуру программы можно представить в виде:

<**программа**>: :=[<номер строки><строка>] ...
<**номер строки**>: := 0|1|2|3|....|9999
<**строка**>: :=[[<оператор>:]...] <оператор> CR/LF
<**оператор**>: := <имя оператора> <тело оператора>

Пример 5.

```
10 COM B%,A(2,179),K%(63)
65 DATA 16.2,277,2224.16:REM  ПАРАМЕТРЫ
71 PRINT "КОДЫ СИМВОЛОВ :";:HEXPRINT A%;:PRINT " В ПЕРЕМЕННОЙ A%"
```

Для оформления и организации программы могут быть использованы такие операторы, как REM — комментарий, STOP — останов, END — оператор конца счета.

Для режима непосредственного счета программа представляет собой одну строку программы без номера.

Алфавит языка. Совокупность символов, используемых для записи и ввода (вывода) в машину программы, данных и команд, составляет алфавит языка.

В алфавит БЕЙСИК входят:

26 прописных букв латинского алфавита;

31 прописная буква русского алфавита (кроме Ё и Ъ);

десятичные цифры 0,1, 2, 3, 4, 5, 6, 7, 8, 9;

шестнадцатеричные цифры 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F;

знаки ! '' % ( ) + — ./: ; <=>? — [\] ↑ # ¤ & '* @

6) символ «Пробел.»

Основные объекты обработки. Машина оперирует с данными двух типов: с числами и символьными данными (с текстом). Эти данные различаются записью, способом кодирования, способом размещения в памяти (форматами), способами обработки. Например, арифметические операции возможны только с числами, а текстовые преобразования— с символьными данными. Вводить с клавиатуры, печатать, записывать и считывать с диска или ленты можно и числа, и символьные данные. В обработке данные используются либо по одному, либо массивами (см. с. 48).

Итак, основными объектами обработки являются:

числовые и символьные константы;

числовые и символьные переменные (в том числе элементы массива);
одномерные и двумерные числовые и символьные массивы.

Константы. Данные могут быть постоянными, т. е. не изменяющимися в процессе всего счета по программе; в этом случае они называются константами. Константы записываются непосредственно в теле оператора программы. Хранятся константы вместе с программой в зоне программы оперативной памяти.

Константы бывают числовыми и символьными.

Формально можно записать:

<**константа**>: :=<числовая константа>|<символьная константа>

Иначе можно записать:

$$\text{<константа>}::=\begin{Bmatrix}\text{<числовая константа>}\\ |\text{<символьная константа>}\end{Bmatrix}$$

Обе записи следует читать одинаково: «константа — это числовая константа или символьная константа», т. е. понятие «константа» является обобщающим и включает понятия «числовая константа» и «символьная константа».

Числовая константа — это число без знака, записанное в естественной или экспоненциальной форме и константа # PI (π), значение которой равно 3,14159265359.

Числовая константа, записанная в естественной форме, может содержать не более 13 цифр (разрядность не более 13). Все разряды могут относиться только к целой части числа (тогда оно записывается как целое), или только к дробной части (тогда перед первой цифрой ставится точка). Кроме того, возможен вариант, когда некоторые разряды относятся к целой, а некоторые к дробной части константы. При этом целая и дробная части константы разделяются точкой. Например:

7	.75	2.5
1624924	.00264	916.57
28	.192769864	12.246875

Запись константы в экспоненциальной форме означает запись в виде мантиссы с разрядностью не более 13 и порядком. Мантисса может быть нормализованной и ненормализованной. Нормализованная мантисса находится в диапазоне 0,1 — 1, т. е. содержит только дробную часть, и первая цифра после точки не равна нулю. Порядок — показатель степени числа 10. Знак отрицательного порядка обозначается символом «—», знак положительного порядка либо опускается, либо обозначается символом «+». Порядок содержит один или два цифровых разряда. Порядок отделяется от мантиссы латинской буквой E.

Например, число 2.5 можно записать в экспоненциальной форме как 2.5 E0 или .25E1, или 25E — 1 и т.д., что соответствует записи $2{,}5\cdot 10^{0}$, $0{,}25\cdot 10^{1}$, $25\cdot 10^{-1}$

Формально числовую константу можно описать в виде:

$$\langle\textbf{числовая константа}\rangle::=\left\{\begin{array}{l}\langle\text{целое}\rangle\ [.\langle\text{целое}\rangle]\ [\text{E}\ [\langle\text{знак}\rangle]\ \langle\text{порядок}\rangle]\\ .\langle\text{целое}\rangle\ [\text{E}\ [\langle\text{знак}\rangle]\ \langle\text{порядок}\rangle]\\ \#\ \text{PI}\end{array}\right\}$$

⟨**целое**⟩ : : = ⟨цифра⟩|...—общее количество цифр не более 13
⟨**знак**⟩ : : = +|—
⟨**порядок**⟩ 55 = |⟨цифра⟩| ⟨цифра⟩

Символьная константа записывается в программе как цепочка символов алфавита языка в кавычках или апострофах: "ИТОГО", "ТАБЛИЦА"; "ПЭКВМ ИСКРА 226", 'КЛУБ «СТРОИТЕЛЬ»'.

Если символьная константа имеет символ-ограничитель в виде кавычек (апострофа), то символ «кавычки» («апостроф») нельзя включать в цепочку информационных символов константы, т е. нельзя писать: «ПЭКВМ «ИСКРА 226», так как второй символ «кавычки» будет воспринят машиной как символ-ограничитель, и следующая пара кавычек тоже станет ограничителями. Нечетное число символов-ограничителей приводит к синтаксической ошибке при наборе.

Следует также помнить, что буквы в кавычках кодируются при наборе HEX-кодами прописных букв, а в апострофах — HEX-кодами строчных букв. Это важно лишь в случае, когда символьное данное сравнивается, участвует в преобразованиях и т. д. Печатается же буква всегда как прописная. Так, в оперативной памяти константа 'ABD' имеет значение 0110 0001 0110 0010 0110 0100, что соответствует HEX-кодам 616264, а константа «ABD» имеет значение 0100 0001 0100 0010 0100 0100, что соответствует HEX-кодам 414244.

Если константа содержит хоть один символ, которого нет на клавиатуре; т. е. символ, не являющийся символом алфавита языка, но участвующий в обработке, то константа записывается в виде HEX-функции, т. е. HEX (<*hh*> <*hh*>...<*hh*>), где *hh* — HEX-код символа.

Формально символьную константу можно описать в виде:

<символьная константа>: : = "<цепочка символов>"|
'<цепочка символов>'|<HEX — функция>
<цепочка символов>: : = {<символ алфавита>} ...

Например, HEX (414243) эта запись соответствует «ABC»; HEX(07).

Переменные. Данные, появляющиеся в машине в процессе счета по программе, т. е. вводимые или получаемые в результате обработки, в отличие от констант хранятся в зоне данных оперативной памяти. В программе вместо этих данных записывают их обозначения в том месте, где данные обрабатываются. Обозначения называются переменными, а данные — их значениями. *Переменная* в языке в отличие от понятия переменной, принятого в математике, обозначает не только данное, но и место в оперативной памяти, где это данное находится. Каждая переменная в языке должна иметь свое имя. Имя переменной иначе называют *идентификатором* переменной. Идентификатором переменной может быть буква латинского алфавита или буква и цифра, при этом цифра следует за буквой. Например, A, A5, X, X9, Y, Y3, M, M6. Всего таким образом можно составить 286 различных идентификаторов переменных.

Если переменная обозначает действительное число, она называет-

ся действительной переменной. В программе действительная переменная записывается в виде идентификатора переменной.

Если переменная обозначает целое число, то она называется целой переменной и записывается в программе идентификатором переменной со значком %: A %, A 2 %, X %.

Если переменная обозначает символьное данное, она называется символьной переменной и записывается в программе идентификатором переменной со значком ¤, т. е. A ¤ , L ¤.

Язык позволяет объединить переменные одного типа в массивы (см. с. 48). Эти переменные имеют один и тот же идентификатор — идентификатор массива и называются элементами массива. Для обозначения переменной — элемента массива в программе кроме идентификатора и значка, соответствующего типу переменной, указывается в круглых скобках один или два параметра, которые определяют местоположение элемента в массиве:

A (5) — пятый элемент действительного массива с идентификатором A;

A (1) — первый элемент того же массива;

X2 % (7) — седьмой элемент целого массива с идентификатором X2;

K ¤ (2,4) — элемент символьного массива с идентификатором K, находящийся на пересечении второй строки и четвертого столбца.

Формально переменные можно описать в виде:

<**переменная**>: ::= <числовая переменная>| <символьная переменная>
<**числовая переменная**>: ::= <простая числовая переменная>|
<элемент числового массива>
<**простая числовая переменная**>: ::= <простая целая>|
<простая действительная>
<**элемент числового массива**>: ::=<элемент целого массива>|
<элемент действительного массива>
<**простая действительная**>: ::= <идентификатор переменной>
<**простая целая**>: ::=<идентификатор переменной> %
<**элемент действительного массива**>: ::=<идентификатор переменной>
(<а.в.>[, <а.в.>])

(**а.в.**) — сокращение от «арифметическое выражение», определяющее местоположение элемента в массиве (номер строки и номер столбца)

<**символьная переменная**>: ::=<простая символьная>|<элемент символьного массива>|<STR — функция>
<**простая символьная**>: ::= <идентификатор переменной> ¤
<**элемент символьного массива**>: ::= <идентификатор переменной> ¤
(<а.в.>[,<а.в.>])
<**STR-функция**>: ::= STR(<символьная переменная> | <метка символьного массива>, <номер символа>[, <число символов>]).
<**номер символа**>: ::= <а.в.>
<**число символов**>: ::= <а.в.>
<**идентификатор переменной**>: ::=<буква латинского алфавита> [<цифра>]

STR-функция обозначает группу подряд расположенных символов в символьной переменной или в символьном массиве, т. е. часть указанной в скобках символьной переменной или символьного массива. Начало группы задается параметром <номер символа>, который обо-

значает порядковый номер символа в символьной переменной или в символьном массиве. При этом учитывается сквозная нумерация символов в символьном массиве без учета деления на элементы.

Число символов группы задается параметром <число символов>.

Пусть значением символьной переменной В ¤ будет ПОВЕРХНОСТЬ, тогда STR (В ¤ , 3,4) будет означать ВЕРХ, т. е.совокупность из четырех символов, начиная с третьего; STR (В ¤ ,7,3) — НОС; STR (В ¤ , 8) — ОСТЬ, т. е. совокупность символов от восьмого символа до конца переменной. Параметр < число символов> в этом случае не указывается.

В оперативной памяти целая переменная в соответствии с форматом целого (10) имеет длину 2 байт, действительная переменная — 8 байт, а символьная переменная может занимать от одного байта до 253 байт. Стандартная (необъявленная в DIM или COM) символьная переменная имеет длину 16 байт

Массивы. Массив — это совокупность пронумерованных переменных одного типа с одинаковыми идентификаторами. Эти пронумерованные переменные являются элементами массива. Массив в программе может быть самостоятельным объектом обработки. Вид записи обращения к массиву в программе зависит от оператора. В операторах DIM и COM используется объявление массива, в остальных операторах (кроме матричных) — метка массива в виде идентификатора переменной с пустыми круглыми скобками.

Матричные операторы имеют свою синтаксическую структуру и свои термины (соответствующие терминам алгебры матриц). Массив в этих операторах обозначается как простая переменная, т. е. идентификатором переменной или идентификатором переменной и одним из значков %, ¤ в соответствии с типом массива.

Массивы бывают одномерные и двумерные. В *одномерных массивах* элементы имеют сквозную нумерацию от 1, т. е. номер элемента одномерного массива совпадает с порядковым номером этого элемента в массиве. В стандартном (необъявленном) одномерном массиве 10 элементов. В *двумерных массивах* элементы разделены на строки (частный случай — одна строка), в каждой строке имеется один или несколько элементов, т. е. один или несколько столбцов. Местоположение элемента двумерного массива задается номером строки и номером столбца, на пересечении которых находится этот элемент. Общее число элементов двумерного массива равно произведению числа строк массива на число столбцов. Нумерация строк и столбцов начинается от 1. Максимально допустимый номер элемента одномерного массива, номер строки и номер столбца двумерного массива — 7999. В стандартном двумерном массиве имеются 10 строк и 10 столбцов.

Формально массив можно описать в виде:

<**массив**> : : <объявление массива>|<метка массива>

<**объявление массива**>: : <объявление числового>|<объявление символьного>

<**объявление числового**>: : <идентификатор переменной> [%] [(<размерность 1> [, <размерность 2>])]

<объявление символьного>: := <идентификатор переменной> ¤ (<размерность1> [, <размерность 2>]) [<длина элемента>]

<размерность 1>: :=<целое> в диапазоне 1—7999, обозначает количество элементов одномерного массива или количество строк двумерного массива

<размерность 2>: := <целое> в диапазоне 1—7999, обозначает количество столбцов двумерного массива

<длина элемента>: := <целое> в диапазоне 1—253. Стандартная длина элемента — 16

<метка массива>: := <метка числового массива>|<метка символьного массива>

<метка числового массива>: := <идентификатор переменной> [%] ( )

<метка символьного массива>: := <идентификатор переменной>¤( )

Пример 6.

```
10 DIM A%(2,2),C(7),K¤(3)60
20 A%(1,1)=7.648:A%(2,1),A%(1,2),A%(2,1),A%(2,2)=2.27
30 INIT (43)K¤(),B¤
40 PRINT K¤(3):PRINT B¤
50 INPUT C(1),C(2),C(3),C(7)
60 MAT PRINT C
70 PRINT C()
```

В строке 10 использованы объявления массивов, в строке 20 — элементы массива А %, в строке 30 — метка массива К¤ и простая символьная переменная В¤, в строке 40— третий элемент массива К¤ и простая символьная переменная В¤, в строке 50 — элементы первый, второй, третий, седьмой массива С.

Запись С (7) и А (2,2), а также К¤ (3) в разных строках обозначают разные объекты.

В строке 60 записан матричный оператор, значит, С — идентификатор матрицы, т. е. обозначает весь массив С, в строке 70 — нематричный оператор и тот же массив записан меткой массива.

Язык позволяет использовать один и тот же идентификатор переменной для переменных и массивов с разными типами данных. Нельзя использовать один идентификатор переменной для двух массивов одномерного и двумерного с переменными одного типа. Например:

К — простая действительная переменная;

К % — простая целая переменная;

К¤ — простая символьная переменная;

К ( ) — метка действительного массива;

К% ( ) — метка целого массива;

К ¤ ( ) — метка символьного массива.

Арифметическое выражение (а. в.). Для обозначения одного числового данного в языке принято обобщающее понятие «арифметическое выражение». Значение арифметического выражения — число. Это может быть, в частности, значение константы, значение числовой переменной, значение функции, результат арифметической операции, в которой участвуют два числа (операнды-слагаемые, сомножители и т.д.), каждое из этих чисел в свою очередь, может быть результатом другой арифметической операции и т. д.

Итак, арифметическое выражение — это числовая константа, числовая переменная или последовательность числовых переменных и числовых констант, разделенных знаками арифметических операций, числовыми функциями, круглыми скобками. В записи арифметического выражения используются также вложенные круглые скобки.

Формально структуру арифметического выражения можно записать в виде:

<а. в.> =:: [{±}] <операнд а.в.> [{<арифметическая операция>
<операнд а.в.>}...]
<операнд а.в.|> : :=<числовая константа>|
<числовая переменная>|<функция>|(<а.в.>)
<функция>: :=<элементарная функция> | <функция пользователя> |
<функция символьной переменной>|<функция округления>
<элементарная функция> : :=<имя функции> (<а.в.>)
<имя функции>: := SIN| COS TAN | ARCSIN| ARCCOS|
ARCTAN | SQR | SGN|RND| LOG|INT|EXP|ABS
<функция пользователя>: :=FN <имя FN>(< а.в.>)
<имя FN > : := <буква латинского алфавита>|<цифра>
<функция округления>: :=ROUND (<а.в.>, <а.в.>)
<
NVAL(**ия символьной переменной**>: :=
символьная переменная>)|

VAL({ <символьная переменная> / <символьная константа> })|

LEN (<символьная переменная>)|

POS (<символьная переменная> <операция сравнения> <операция сравнения> { ''<цепочка символов>'' / <HEX — код> / <символьная переменная> })

<арифметическая операция>: :=<сложение>|
<вычитание> |<умножение>| <деление> | <возведение в степень>
<сложение>: :=+
\/ычитание>: :=—
<умножение>: :=*
<деление> : :=/
<возведение в степень> : :=∧ (на печати это символ ↑)
<операция сравнения> : :=<меньше>| <меньше или равно> | <равно>|
<больше>|<больше или равно>|<не равно>
<меньше> : :=<
<меньше или равно> : :=< =
<равно> : :=
<больше или равно> : : = > =
<не равно> : :=< >

Примеры записи арифметических выражений:

76. 241.
K %
A+SIN (X — 0.5)
Y (5.6)
ARCTAN (Y)
A* (—X)+5.678
(A*X∧2+B*X+C)/SIN (SQR (X))
(FNK (X—7.548*EXP (X)))*2.5

Арифметические операции всегда являются частью арифметического выражения.

Порядок вычисления арифметического выражения слева направо с учетом приоритетности арифметических операций.

Сначала определяется значение выражения в скобках. Если в арифметическом выражении использованы вложенные скобки, то вычисление начинается с внутренних скобок. Первыми вычисляются значения функций и выполняется возведение в степень, затем умножение, деление и последними выполняются сложение и вычитание.

Общая характеристика операторов. Оператор — это элемент строки программы. По назначению, способу исполнения, синтаксической конструкции операторы можно объединить в отдельные группы.

Основные операторы, используемые для организации программы и управления счетом:

REM — оператор-комментарий;

CLEAR — оператор очистки памяти;

RUN — оператор запуска программы;

STOP — оператор останова;

END — оператор окончания счета;

TRACE — оператор трассирования;

RENUMBER — оператор перенумерации строк программы;

DIM, COM, COMCLEAR, DATA, DEFFN, DEFFN′ — операторы объявления и задания данных;

LET, READ, INIT — операторы, изменяющие значение переменных;

GOTO, IF THEN, ON, ON ERROR — операторы программных переходов;

GOSUB, GOSUB′, RETURN, RETURN CLEAR — операторы организации обращения к подпрограммам;

FOR, NEXT — операторы цикла;

SELECT—оператор задания устройств ввода-вывода, паузы индикации, единицы измерения тригонометрических функций.

Операторы побитных преобразований и логических операций:

ADD — оператор двоичного сложения;

BIN — оператор преобразования десятичного числа в двоичное;

AND, OR, XOR, BOOL — операторы, реализующие логические операции.

Операторы преобразования данных:

CONVERT — оператор преобразования данного из числового формата в символьный и наоборот;

PACK, UNPACK — операторы упаковки и распаковки чисел по формату;

¤PACK, ¤UNPACK операторы упаковки и распаковки данных;

ROTATE оператор циклического сдвига битов каждого байта;

¤TRAN оператор преобразования кодов.

Матричные операторы:

MAT REDIM — оператор переопределения размерности массива;

MAT READ, MAT=, MAT ZER, MAT CON, MAT IDN—операторы присвоения элементам массива соответствующих значений;

MAT INPUT — оператор ввода значений элементов массива;

MAT PRINT — оператор печати массивов;

MAT +,MAT —,MAT*, MAT ( )*, MAT INV, d, MAT TRN — операторы, реализующие основные операции алгебры матриц (сложение, вычитание, умножение матриц, умножение матрицы на скаляр, инвертирование и транспонирование матрицы).

Операторы сортировки:

MAT CONVERT — оператор преобразования числового массива к виду, удобному для сортировки;

MAT SORT, MAT MERGE — операторы получения массива-локатора;

MAT MOVE — оператор, выполняющий перепись исходного массива с упорядочением значений элементов в соответствии с массивом-локатором в результирующий массив;

MAT COPY — оператор, выполняющий побайтную перепись информации из одного символьного массива в другой символьный массив;

MAT SEARCH — оператор выделения символов, удовлетворяющих условию.

Операторы ввода-вывода:

INPUT, LINPUT, KEYIN — операторы ввода данных с клавиатуры;

PRINT, PRINTAT, HEXPRINT, PRINTUSING — операторы вывода данных;

LIST — оператор вывода текста программы и справочной информации о программе на БОСГИ и АЦПУ;

DATA LOAD BT, DATA SAVE BT — универсальные операторы ввода-вывода информации;

PLOT — оператор вывода информации на графические устройства;

% — формат для PRINTUSING;

IF END THEN-оператор анализа на конец считываемого файла и выполнения условного перехода в соответствии с результатом анализа.

Дисковые операторы:

DATA SAVE DC OPEN, DATA LOAD DC OPEN, DATA SAVE DC, DATA LOAD DC — операторы, обеспечивающие вывод данных в файл и считывание данных из файла;

DATA SAVE DC CLOSE — оператор закрытия файла данных на диске;

DBACKSPACE, DSKIP — операторы перемещения в файле;

SAVE DC, LOAD DC — операторы вывода и загрузки программы.

MOVE, COPY, VERIFY-операторы копирования и контроля диска;

SCRATCH — оператор ликвидации файла;

SCRATCH DISK — оператор, выполняющий каталогизацию диска;

MOVE END — оператор, изменяющий область каталога файлов;

LIST DC, LIMITS — операторы получения справочной информации (в непосредственном счете и в программе) о файлах на диске.

Дисковые операторы прямой адресации:

DATA SAVE DA — операторы вывода и ввода данных;

LOAD DA, SAVE DA — операторы вывода и ввода программы.

Оператор ¤ GIO′ — специальный оператор, позволяющий включать в программу БЕЙСИК фрагменты программы, составленной в машинных инструкциях.

Оператор ¤ GIO — обобщенный оператор обмена, позволяющий включать фрагменты программы, составленной в командах канального процессора.

Операторы работы с КНМЛ:

DATA LOAD, DATA SAVE, DATA RESAVE — операторы ввода-вывода данных;

SKIP, BACK SPACE, REWIND — операторы подмотки (вперед, назад) и перемотки ленты;

LOAD, SAVE — операторы ввода-вывода программы.

Глава VI

РАБОТА В РЕЖИМЕ НЕПОСРЕДСТВЕННОГО СЧЕТА

1. ОБЩИЕ СВЕДЕНИЯ

В режиме непосредственного счета может быть выполнено большинство операторов языка. Не выполняются в режиме непосредственного счета операторы DATA, %, DEFFN, DEFFN′, ON ERROR, так как эти операторы могут восприниматься и иметь смысловое значение лишь в режиме счета по программе. Не выполняются команды программных переходов, кроме GOTO, задающего начало отладочного режима. Некоторые операторы, например, LOAD DC, LOAD DA, LOAD, в режиме непосредственного счета выполняются иначе, чем в режиме счета по программе. Наиболее часто в непосредственном счете используются операторы диалогового управления машиной: LIST DC, LIST, LOAD DC, SAVE DC, CLEAR, RUN, SELECT, RENUMBER (3), а также операторы: PRINT, LET, FOR, NEXT.

Оператор PRINT занимает особое место. Он как бы переключает машину в режим калькулятора, т. е. позволяет вычислить значение арифметического выражения, вызвать для просмотра на экран значение любой переменной, значения массива, не изменяя при этом значений переменных, массивов и текста программы, находящейся в зоне программы памяти. Числовые и символьные константы используются по правилам, принятым в языке.

Приведем примеры набора в НС оператора печати:

1) : PRINT 2.748261+0.0791624

Результат выполнения оператора — индикация суммы чисел;

2) : PRINT SQR (0.25)

Результат — индикация значения корня от числа 0.25

3) : PRINT SQR (X)

Результат — индикация значения корня от числа, являющегося значением переменной X. Значение этой переменной осталось неизменным.

4) : PRINTX — индикация значения переменной X.

5): PRINT "ЗНАЧЕНИЕ АРИФМЕТИЧЕСКОГО ВЫРАЖЕНИЯ="; 5.2∧12+5.2—7.5E6

В результате выполнения оператора проиндицируются значение символьной константы и значение арифметического выражения.

Набираются числа по тем же правилам, по которым константы записываются в языке, т. е. в естественной и экспоненциальной форме с нормализованной или ненормализованной мантиссой. Знак «+» перед положительным числом, а также перед положительным порядком и перед положительной мантиссой числа экспоненциальной формы записывать и набирать не обязательно. Не набираются также незначащие нули в целой и дробной частях числа и мантиссы числа.

Примеры набора чисел;

1) 7,510	7.510	или 7.51	и т. д.
2) $16{,}24 \cdot 10^{12}$	16.24E12	или. 1624E14	и т. д.
3)—0,005	—.005	или—.5E—2	и т. д.

Индицируется число после выполнения оператора PRINT в естественной форме, если оно по модулю не больше 10^{13} и не меньше 0.1, иначе — в экспоненциальной форме с нормализованной мантиссой.

Знак положительного числа и положительной мантиссы числа экспоненциальной формы индицируется символом «пробел», знак положительного порядка не индицируется.

Примеры индикации числа оператором PRINT.

1) : PRINT + 7.51
   7.51
2) :PRINT —.005
   —.5E—2
3): PRINT + 16.175E—1
   1.6175

Использовать в строке непосредственного счета операторы (кроме PRINT), содержащие переменные, следует с большой осторожностью, так как значения переменных изменяются. Если действия, выполняемые в режиме непосредственного счета, не относятся к программе, находящейся в памяти, то пользоваться можно лишь переменными, которые не задействованы в программе.

Например, в программе используется переменная X, а в непосредственном счете необходимо вычислить значение

$$F(X) = X^3 - 5{,}7X^2 + X - 7{,}64X^{-1} + 277 \text{ для } X = 276{,}9888241.$$

Набирать многократно многоразрядное значение X неудобно, поэтому для набора необходимо воспользоваться переменной, которая не задействована в программе (K) и оператором присвоения значения (LET):

```
K=276.988241 : PRINT K∧3—5.7 * K∧2+
K—7.64 * K∧ (—1)+277.
```

Если же возникла необходимость изменить значение переменной, используемой в программе, то следует выполнить оператор присвоения значения (LET) в режиме непосредственного счета, используя ту переменную, значение которой нужно изменить.

Например, при выполнении счета по программе произошел останов по сбою (ERR 03 — деление на 0) ввиду неудачного задания параметра счета экспериментатором-пользователем. Экспериментатор-пользователь, проанализировав причину ошибки, решил, что счет нужно продолжить, изменив предварительно значение переменной С % (200 — новое значение С %). Для реализации этого решения необходимо в непосредственном счете выполнить оператор С % = 200 и команду CONTINUE.

Подробнее о выполнении операторов PRINT см. на с. 85, LET — на с. 78, о вводе строки непосредственного счета — на с. 30, о редактировании строки непосредственного счета на с. 32—33.

Итак, строка непосредственного счета набирается без номера и используется сразу же при нажатии на клавишу CR/LF. Набор возможен лишь в режиме ожидания (индикация : —), т. е. когда счет по программе приостановлен или когда зона программы памяти пуста.

2. ИСПОЛЬЗОВАНИЕ ЭЛЕМЕНТАРНЫХ ФУНКЦИЙ

Функции, используемые в арифметических выражениях, вычисляются с округлением до 13-й значащей цифры мантиссы, точность же вычислений функций различна и для большинства функций (особенно тригонометрических) зависит от значения аргумента.

Функции ABS, LOG, SQR, EXP, SGN, INT.

Рассмотрим структуру названных элементарных функций:

ABS (<а.в.>) — абсолютное значение арифметического выражения, являющегося аргументом функции;

Пример 7.

```
:PRINT ABS(-16.241)
 16.241

:X=-2:PRINT 7.5+ABS(X*3)
 13.5
```

LOG (<а.в.>) — логарифм натуральный арифметического выражения;

Пример 8.

```
PRINT LOG(2/.75)
 .9808292530119
```

SQR (<а.в.>) — квадратный корень от значения арифметического выражения;

Пример 9.

```
:PRINT SQR(144.6/47)
 1.754022732038
```

EXP (<а.в.>) — число, равное числу E в степени значения арифметического выражения. Допустимые значения <а.в.> не более 230;

Пример 10.

```
PRINT EXP(1)
 2.718281828459
```

— значение числа E

SGN (<а.в.>) — знак арифметического выражения, т. е. значение функции равно:

1, если значение арифметического выражения > 0;

0, если значение арифметического выражения $= 0$;

— 1, если значение арифметического выражения < 0;

Пример 11.

```
:
*PRINT SGN(2.5),:PRINT SGN(-7),:PRINT SGN(5-5)
 1             -1             0
```

INT (<а.в.>) — наибольшее целое число, ближайшее к значению арифметического выражения (с учетом знаков), но не превосходящее это значение.

Пример 12.

```
*PRINT INT(6.3),:PRINT INT(6.8),:PRINT INT(-6.3),:PRINT INT(6)
 6             6             -7            6
```

Функция RND датчик случайных чисел.

RND (<а.в.>) — случайное число в диапазоне 0 — 1. Значение функции однозначно зависит от аргумента только тогда, когда равен нулю аргумент первой вычисляемой функции RND и вычисление функции —первое после запуска программы на счет оператором RUN. Если в программе многократно вычисляется функция RND от нуля и ни разу не вычисляется от ненулевого аргумента, то совокупность вычисляемых последовательно значений функции будет постоянной. Эта последовательность получается всякий раз, когда счет начинается с начала программы. Это очень удобно для отладки программы.

Если аргумент вычисляемой функции RND не равен нулю, то получить повторяющуюся совокупность случайных чисел практически нельзя. Эта особенность функции RND относится лишь к режиму счета по программе.

Пример 13.

```
10 PRINT RND(0)
20 FOR K=1TO5:PRINT RND(1):NEXT K
30 PRINT
```

```
 9.25059814E-02     9.25059814E-02
 .7758946659324     .5469689496022
 .4825773718396     .3254545736245
 .8386852927801     .3742312738384
 .3167427138641     .2855840332378
 .1187966219862     .7302589006817
```

Функция ROUND. Ее называют функцией округления.

ROUND (<а.в.>, <а.в.>) — число, равное округленному значению первого арифметического выражения. Параметр округления задается значением второго арифметического выражения. Если значение это имеет дробную часть, то она автоматически отбрасывается. Итак, параметр округления — целое число K. Округление выполняется до:

K-го десятичного знака, если $K > 0$;

до ближайшего целого, если $K = 0$;

до ($|K| + 1$)-го знака в целой части, если $K < 0$.

Пример 14.

```
:PRINT ROUND(2.341,2),:PRINT ROUND(2.347,2)
 2.34              2.35

:PRINT ROUND(-22.341,0),:PRINT ROUND(22.7,0)
-22                23

:PRINT ROUND(175,-2),:PRINT ROUND(175,-1)
 200               180

*PRINT ROUND(175,-3),:PRINT ROUND(700,-3)
 0                 1000
```

Тригонометрические функции. К тригонометрическим функциям языка относятся: SIN, COS, TAN, ARCSIN, ARCCOS, ARCTAN. При вычислении значений этих функций угловые величины могут быть заданы в радианах, в градусах и градах (2π радиан = 360 градусов 400 градов). Системой БЕЙСИК для угловых величин в тригонометрических функциях единицей измерения автоматически выбираются радианы.

Для задания угловых величин в градусах необходимо выполнить оператор SELECTD, в градах SELECTG, а для возврата к радианам

— SELECTR. Установленная единица измерения углов будет относиться ко всем вычисляемым тригонометрическим функциям до тех пор, пока не будет установлена новая единица измерения или выполнен оператор CLEAR.

Пример 15.

```
:PRINT 2*SIN(4PI/3):REM  УГОЛ В РАДИАНАХ
 1.732050807569

:SELECT D:PRINT COS(90).:PRINT TAN(12):REM  УГЛЫ В ГРАДУСАХ
 0                .21255656167

:SELECT R:PRINT TAN(12).:PRINT ARCCOS(.578):REM  УГЛЫ В РАДИАНАХ
-.6358599286616  .9545206395577
```

Глава VII
РАБОТА В РЕЖИМЕ СЧЕТА ПО ПРОГРАММЕ

1. ОБЩИЕ СВЕДЕНИЯ

Режим счета по программе — основной режим работы, используемый для решения задач во всех ЭВМ. Он предусматривает предварительное создание программы, ввод ее в машину и исполнение, т. е. счет по программе. Один раз созданная программа может исполняться многократно. Программы для разового исполнения создаются редко, например, когда требуется выполнить разовый расчет с высокой точностью.

Процесс создания программы принято называть программированием. Программирование — понятие емкое. Оно обозначает всю совокупность действий, выполняемых при создании рабочей программы. Эти действия можно разделить на этапы:

1) постановка задачи;
2) алгоритмизация;
3) непосредственное программирование;
4) отладка программы;
5) оформление инструктивной и отчетной документации.

2. ПОСТАНОВКА ЗАДАЧИ

Первый этап можно назвать подготовительным. Характер действий, выполняемых на этом этапе, определяется типом задачи, которую предстоит решить, а цель действий общая — выдача задания на программирование.

Для задач учетного характера, планово-экономических, бухгалтерских, задач по сбору и обработке информации, используемой для

управления производством, выделяется участок учета. Определяется вся совокупность информационных потоков, связывающих отдельные объекты участка учета, т. е. объем, структура исходной, промежуточной информации, а также результатов обработки, выдаваемых заказчику. Под структурой исходной информации понимается вид исходных документов, количество и назначение реквизитов документа, разрядность числовых и символьных данных, участвующих в обработке. Аналогично под структурой результирующей (выходной) информации понимается описание выходных документов, выдаваемых на печать — табличных документов, а также структура информации на техническом носителе, если получение таковой предусматривается в задаче.

Определяется способ обработки данных, выявляются логические и балансовые связи реквизитов документа, которые используются для контроля введенных данных. Такие задачи относятся к информационным. Математический аппарат в них используется слабо, практически применяются лишь арифметические операции.

Для научных, инженерно-технических и экспериментальных задач подготовительная работа сводится к определению исходных данных, т. е. к определению их количества и области возможных значений, а также к определению функциональных зависимостей между исходными данными, логических связей и формы выдаваемой информации.

Постановка задачи — ее основа, фундамент, поэтому постановщики задачи должны обладать высоким уровнем квалификации в той области, к которой относится решаемая задача.

3. АЛГОРИТМИЗАЦИЯ

Цель второго этапа — создание алгоритма решения задачи.

Алгоритм — это точное предписание, определяющее процесс преобразования исходных данных в искомый результат. Алгоритм должен обладать следующими свойствами:

определенностью, или точностью, не оставляющей места для произвола, и дискретностью, т. е. представлением в виде совокупности элементарных предписаний — дискретных шагов;

массовостью, под которой понимается возможность получения искомого результата для любой совокупности исходных данных из области возможных значений;

результативностью, т. е. в алгоритме обязательно должны быть такие предписания, которые при выполнении преобразований исходных данных точно определяют, что является результатом и при каких условиях процесс преобразования следует остановить.

Алгоритмом можно назвать любое правильно составленное методическое предписание, инструкцию. Эти алгоритмы носят описательный характер, и элементарным предписанием в них является описание одного действия из всей совокупности, включенной в алгоритм.

При составлении алгоритма задачи руководством к действию является постановка задачи — задание на программирование. Для инженерно-технических и других задач, использующих для счета матема-

8. Символы в схемах алгоритмов

Символ	Содержание	Символ	Содержание	Символ	Содержание
	Процесс		Ручной ввод		Дисплей
	Решение		Ввод-вывод		Линия потока
	Модификация		Документ		Пуск-останов
	Ручная операция		Магнитная лента		Межстраничный соединитель
	Сортировка		Магнитный диск	—	—

тический аппарат, не ограничивающийся арифметическими операциями и вычислениями элементарных функций, предварительно следует выбрать численный метод и обосновать его выбор, а затем приступить к составлению алгоритма. Алгоритм может носить описательный характер, что не очень удобно и практически не используется. Чаще всего алгоритм записывается в виде блок-схемы. При составлении блок-схемы алгоритма необходимо использовать стандартные обозначения, определенные по ГОСТ 19.003—80. Эти символы вслед за развитием вычислительной техники, теории алгоритмов и языков программирования изменяются, но основные символы остаются неизменными (табл. 8).

Одна и та же задача может быть решена несколькими способами, одна и та же постановка задачи может быть реализована различными алгоритмами.

4. НЕПОСРЕДСТВЕННОЕ ПРОГРАММИРОВАНИЕ

На этом этапе записывается алгоритм решения задачи на входном языке машины в соответствии с предусмотренной в языке структурой программы. Для БЕЙСИК-программы, т. е. программы пользователя, записанной на языке БЕЙСИК, используется строчная структура.

Программа — совокупность пронумерованных строк, состоящих из одного или нескольких операторов языка, разделенных символом «двоеточие» (см. с. 43). Одному и тому же алгоритму могут соответствовать несколько программ, дающих правильное решение.

Для выбора наилучшего алгоритма и наилучшего варианта программы пользуются различными критериями. Программа, выдающая результат наиболее быстро, всегда считается лучшей. Для больши́х информационных задач временной критерий оценки часто отступает на второе место, а основным критерием становится эффективность использования оперативной памяти, технических носителей и технических средств.

Написанная программа не является еще рабочей, так как в ней возможны ошибки, поэтому на следующем этапе выполняется отладка программы, т. е. выявление и устранение всех ошибок, которые были допущены при написании постановки задачи, при создании алгоритма, при написании программы и при вводе ее в машину. Методика выполнения отладки рассмотрена ниже (см. с. 113).

Пример 16.

Вычислить значение функции $F(X) = AX^2 + BX + C$ для $X = 1.1,\ 2.2$ при $A = 2,\ B = 4,\ C = 8$.

а) В непосредственном счете такая задача решается выполнением двух операторов PRINT. Они могут быть набраны в одной строке или двумя строками:

```
:PRINT 2*1.1↑2+4*1.1+8:PRINT 2*2.2↑2+4*2.2+8
;
```

```
:PRINT 2*1.1↑2+4*1.1+8
:PRINT 2*2.2↑2+4*2.2+8
!
```

```
 X
```

```
:X=1.1:PRINT 2*X↑2+4*X+8
:X=2.2:PRINT 2*X↑2+4*X+8
!
```

В программном варианте 3 вторую строку можно получить редактированием первой строки после перевызова ее, т. е. нажать EDIT RECALL и заменить 1.1 на 2.2.

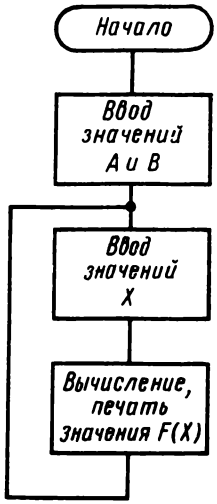
Блок-схема алгоритма приведена на рис. 8.

б) В режиме счета по программе эта задача может быть решена двумя способами: как разовая, т. е. с фиксированными значениями коэффициентов A, B, C, и значениями X, и как универсальная, позволяющая получить значение функции для любой допустимой совокупности действительных значений A,B,C,X.

```
10 DEFFN F(X)=2*X↑2+4*X+8
20 PRINT FNF(1.1):PRINT FNF(2.2)
```

←

Рис. 8. Блок-схема алгоритма вычисления $F(x)$ в непосредственном счете



→

Рис. 9. Блок-схема алгоритма вычисления $F(x)$ в режиме счета по программе

```
10 INPUT "ВВЕДИТЕ ЗНАЧЕНИЯ ПАРАМЕТРОВ A,B",A,B
20 INPUT "ВВЕДИТЕ ЗНАЧЕНИЕ X",X
30 PRINT A*X↑2+B*X+C:GOTO 10
```

Запуск на счет происходит при нажатии на клавиши RUN CR I.F.

Блок-схема алгоритма приведена на рис. 9.

Последняя программа не имеет естественного завершения счета, так как она допускает любое переменное количество вычисляемых значений и прервать ее выполнение можно PESET в момент ожидания ввода данных, т. е. при индикации ВВЕДИТЕ ЗНАЧЕНИЯ ПАРАМЕТРОВ?

5. ОПЕРАТОРЫ, ИСПОЛЬЗУЕМЫЕ ДЛЯ ОРГАНИЗАЦИИ ПРОГРАММЫ

Оператор CLEAR. Оператор очистки памяти имеет следующую структуру:

CLEAR [<параметр CLEAR>]
<**параметр CLEAR**> : : = N|V|P [<диапазон строк>]
<**диапазон строк**> : : = <номер строки 1> [, <номер строки 2>].
<номер строки 2>

Оператор CLEAR предназначен для очистки оперативной памяти полностью или частично, при этом стираемая часть задается соответствующим параметром CLEAR:

CLEAR без параметров очищает всю память;

CLEAR N стирает значения необщих переменных.

Текст программы и значения общих переменных, объявленных ранее в COM, сохраняются.

CLEAR V стирает все значения переменных (общих и необщих). Текст программы сохраняется в оперативной памяти.

CLEAR P без указания номеров строк стирает весь текст программы, но при этом описания и значения переменных сохраняются.

Часть стираемой программы указывается диапазоном номеров строк:

<номер строки 1> — номер первой стираемой строки. Если он опущен, то программа стирается с начала, т. е. с первой записанной в программе строки.

<номер строки 2> — номер последней стираемой строки в диапазоне. Если номер опущен, то программа стирается до конца. Оператор в основном используется в непосредственном счете.

Приведем примеры синтаксической структуры оператора:

1) : CLEAR — очищается вся память;
2) : CLEARV — стираются все значения переменных;
3) : CLEAR N — стираются значения необщих переменных;
4) : CLEAR P — стирается вся программа;
5): CLEAR P : CLEAR N — стираются вся программа и необщие данные;
6): CLEAR P40 — стирается часть программы, начиная со строки 40 до конца программы;
7) : CLEAR P 40, 90 — стирается часть программы со строки 40 до строки 90 включительно;
8): CLEAR P, 90 — стирается часть программы с начала до 90-й строки включительно.

Оператор RENUMBER. Оператор перенумерации имеет следующую структуру:

RENUMBER [<номер строки 1>] [, <новый параметр>]
<**новый параметр**> : := <номер строки 2> [, <шаг>]
<**шаг**>: := <цифра> | < цифра> <цифра>

Оператор RENUMBER предназначен для перенумерации строк программы и ссылок на них в программе. Оператор может использоваться без параметров и с параметрами:

<номер строки 1> — номер строки, начиная с которой выполняется перенумерация всех строк до конца программы. Если номер не указан в операторе, то перенумеровываются все строки с начала программы. Ссылки перенумеровываются во всей программе независимо от наличия параметра <номер строки 1>.

<номер строки 2> — новый номер, присваиваемый строке, с которой начинается перенумерация. Если параметр <номер строки 2> не указан, то новый номер строки равен значению шага.

<шаг> — шаг перенумерации, задаваемый числом в диапазоне 1—99. Если шаг не указан, то он равен 10.

Приведем примеры синтаксической структуры оператора:

1) RENUMBER — программа получает нумерацию строк с шагом 10, первая новая строка 10;
2) RENUMBER 70, 115,5 — строка 70 получает номер 115, номера последующих строк с шагом 5 (120, 125, 130 ...);
3) RENUMBER 70, 110 — строка 70 получает номер 110, номера последующих строк с шагом 10 (120, 130, ...);

4) RENUMBER 70 — перенумерация как оператором RENUMBER, 70, 10, 10 (см. п. 2);

5) RENUMBER 70,, 5 — как оператором RENUMBER 70, 5,5 (см. п. 2);

6) RENUMBER, 20 — первая строка программы получает номер 20, последующие строки нумеруются с шагом 10;

7) RENUMBER, 20, 5 — первая строка программы получает номер 20, последующие нумеруются с шагом 5;

8) RENUMBER, , 20 — как оператором RENUMBER, 20, 20 (см. п.7).

Оператор в основном используеся в режиме непосредственного счета.

Оператор RUN. Оператор имеет следующую структуру:

RUN [<номер строки>]

Оператор предназначен для запуска программы на счет. Оператор используется с параметром и без параметра. Выполнение оператора зависит от наличия параметра:

1) параметр отсутствует. Выполняется начальный запуск, при котором перед выполнением первого оператора строки программы (с меньшим номером) исследуется формальная правильность загруженной программы. Значения общих переменных не изменяются, а необщим числовым переменным присваивается значение 0, необщим символьным переменным присваивается значение в виде совокупности пробелов по всей длине символьной переменной, т. е. каждый символ — пробел (HEX—код 20). Устанавливается в единицу указатель текущего значения константы в операторе DATA, используемого оператором READ;

2) параметр задан. Начинается счет по программе с первого оператора указанной строки. Значения всех переменных и указателя констант в DATA сохраняются. Используется оператор RUN <номер строки > для продолжения счета после прерывания счета. Номер, указываемый в операторе, не должен находиться внутри цикла или подпрограммы.

Если программа начинается со строки 10, то из сказанного следует, что RUN10 и RUN выполняются по-разному.

Оператор STOP. Оператор останова счета имеет следующую структуру:

STOP [<символьная константа>]

Оператор STOP предназначен для программируемого останова машины при счете по программе. Оператор STOP используется с параметром и без параметра. Параметр задается символьной константой и содержит инструктивную информацию для пользователя.

При выполнении оператора STOP на экране индицируется слово STOP и значение символьной константы, если она была указана в операторе, после чего происходит останов машины. Устанавливается режим ожидания (: -). Для возобновления счета по программе дальше достаточно нажать на клавишу CONTINUE.

Используется оператор при отладке и при счете, чтобы дать пользователю возможность выполнить ручную операцию: установить диск,

снять защиту записи, сменить диск, заправить бумагу, установить страницу и т. д.

Оператор STOP в непосредственном счете не используется.

Приведем примеры записи оператора

```
1) 170 STOP
2) 220 STOP "СНИМИТЕ ЗАЩИТУ ЗАПИСИ"
```

Оператор END. Оператор END означает конец счета по программе. После выполнения оператора на экране индицируется:

```
END PROGRAM
FREE SPACE,  XXXXX
```

где XXXXX — пятизначное число, равное числу байтов свободной памяти.

Счет по программе прекращается. Использование END в конце программы, т. е. последним записанным оператором программы не обязательно.

В то же время в программе можно использовать несколько операторов END, если программа заканчивается на разных ветвях. Такое построение программы не совсем логично.

В непосредственном счете оператор END позволяет в любое время (в режиме ожидания :_) индицировать свободную память. Свободная память до счета будет больше свободной памяти после счета.

Пример 17.

```
10 INPUT X
20 Y=SIN(X)+5.748567
30 PRINT "X=";X,"Y=";Y
40 END
```

Оператор REM. Оператор-комментарий имеет следующую структуру:

REM [<цепочка символов>]

Оператор REM предназначен для включения в текст программы различных пояснений (комментария) в виде любой совокупности символов, кроме двоеточия (:). Оператор REM не исполняется машиной, т. е. при счете никакой индикации нет, а при выводе текста программы оператором LIST он индицируется как любой другой. В памяти машины оператор REM хранится, увеличивая тем самым объем памяти, используемый программой. В непосредственном счете не используется.

Пример 18.

```
70 REM НАЧАЛО СОРТИРОВКИ
120 K=1+M:REM ИЗМЕНЕНИЕ ПАРАМЕТРА
```

Оператор SELECT. Оператор выбора УВВ и режима его работы при обмене имеет следующую структуру:

SELECT ⟨список SELECT⟩

⟨**список** SELECT⟩ : : = ⟨параметр SELECT⟩ [{, параметр SELECT⟩}...]

⟨**параметр** SELECT⟩ : : =
- CI ⟨ФАУ⟩
- CO ⟨ФАУ⟩ [(⟨длина строки⟩)]
- TAPE ⟨ФАУ⟩ [(⟨длина строки⟩)]
- DISK ⟨ФАУ⟩ { F / R } [(⟨длина строки⟩)]
- PLOT ⟨ФАУ⟩ [(⟨длина строки⟩)]
- LIST ⟨ФАУ⟩ [(⟨длина строки⟩)]
- PRINT ⟨ФАУ⟩ [(⟨длина строки⟩)]
- # ⟨логический номер⟩ ⟨ФАУ⟩ [{ F / R }]
- P [⟨цифра⟩]
- D
- G
- R

⟨длина строки⟩ — целое число из диапазона 1 —7999.

⟨ФАУ⟩ — физический адрес устройства (**HEX**-код).

Оператор SELECT предназначен для:

выбора параметров и адресов устройств ввода-вывода при выполнении обмена соответствующими операторами ввода-вывода;

выбора адресов устройств для консольного ввода-вывода;

выбора единицы измерения угловой величины при вычислении тригонометрических функций;

выбора физического адреса устройства ввода-вывода, соответствующего данному логическому номеру.

Оператор SELECT содержит один параметр SELECT или несколько параметров SELECT, разделенных символом запятая. Параметр SELECT может принимать одно из значений, перечисленных в фигурных скобках.

Оператор SELECT является как бы директивным, т. е. сам в обмене с устройствами ввода-вывода не участвует, а лишь указывает БЕЙСИК-системе, с какого устройства выполнить ввод (вывод), если встретится при исполнении программы соответствующий оператор ввода-вывода, и какие параметры устройства (длину строки, скорость обмена) следует при этом использовать. Данное указание система запоминает и применяет всякий раз при исполнении ввода-вывода до тех пор, пока действие параметра оператора SELECT не будет отменено. Аналогично и задание единицы измерений угловых величин действует до отмены действия параметра SELECT.

Итак, параметр SELECT действует до тех пор пока не будет выполнен другой оператор SELECT с тем же параметром SELECT, имеющим другое значение ФАУ или длину строки;

выполнен CLEAR без параметров, устанавливающий начальные значения параметров CO, CI и т. д.;

нажата кнопка системного сброса (SR).

Все параметры SELECT независимы (т. е. SELECT LIST отменяет лишь ранее выполненный SELECT LIST) и не влияют на действие параметра PRINT (т. е. не отменяют выполненного ранее оператора SELECT PRINT).

Оператор используется в равной степени как в непосредственном счете, так и в программе, когда необходимо изменить ФАУ устройства ввода-вывода, или длину строки, или то и другое, т. е. когда необходимо отказаться от стандартных (начальных) значений параметров или, наоборот, установить начальные значения параметров устройств.

Назначение оператора SELECT с конкретными значениями параметров SELECT:

SELECT CI < ФАУ> — задает устройство для консольного ввода, т. е. для ввода диалоговой информации. Система БЕЙСИК устанавливает для CI клавиатуру с ФАУ 01. Используется оператор редко, например при работе с клавиатурой как с периферийным устройством ввода;

SELECT CO < ФАУ> [(<длина строки>)] задает устройство для консольного вывода, т. е. для вывода диалоговой информации (набираемой с клавиатуры и выводимой системой в виде сообщений об ошибках, информации операторов STOP, END и т. д.).

БЕЙСИК-система устанавливает для консольного вывода, а также для вывода операторами, соответствующими параметрам SELECT: PRINT и LIST, БОСГИ символьный с ФАУ 05 и длиной строки 80 символов. Для получения протокола работы машины на бумаге необходимо выполнить:

SELECT CO 0C — длина строки сохранится (80 символов)

SELECT CO 0C (65) — на печати установится длина строки в 65 символов. Для возврата к БОСГИ нужно выполнить либо CLEAR, либо SELECT CO 05 (80);

SELECT PRINT <ФАУ > [(<длина строки>)] задает устройство вывода и длину строки вывода информации операторами PRINT, PRINTUSING, HEXPRINT, MAT PRINT.

Например:

SELECT PRINT 18 — НГМД;

SELECT PRINT 0C (130) — печатающее устройство с длиной строки 130 символов;

SELECT LIST <ФАУ > [(<длина строки>)] задает устройство и длину строки вывода текста программы и справочной информации операторами LIST и LIST DC:

SELECT LIST 0C:LIST — распечатка текста программы оператором LIST, для которого задано печатающее устройство;

SELECT TAPE <ФАУ> [(<длина строки>)] задает устройство ввода-вывода параметром ФАУ и размер физической записи парамет-

ром <длина строки > для ленточных (и универсальных) операторов ввода-вывода SAVE, LOAD, DATA SAVE BT, DATA LOAD BT и т. д. Для параметра TAPE система устанавливает ФАУ 08;

SELECT DISK < ФАУ > $\begin{Bmatrix} F \\ R \end{Bmatrix}$ [(<длина строки>)] задает дисковое устройство для ввода-вывода. БЕЙСИК-система устанавливает НГМД с ФАУ 18F.

SELECT DISK 18 R — НГМДR;

SELECT DISK 1CF — HMDF;

SELECT PLOT < ФАУ> [(<длина строки>)] задает графическое устройство для вывода оператором PLOT. БЕЙСИК-система устанавливает БОСГИ графический с ФАУ 10. Длина строки — объем буфера вывода. Например:

SELECT PLOT 14 (7999) — графопостроитель Н-306;

SELECT # <логический номер> <ФАУ> $\left[\begin{Bmatrix} F \\ R \end{Bmatrix}\right]$ устанавливает соответствие между логическим номером и ФАУ. Этот оператор допускает использование списка параметров. Логический номер, задаваемый цифрой от 0 до 7 включительно, пользователь выбирает для обозначения того или другого устройства. Нельзя присваивать разным ФАУ одновременно одинаковые логические номера, но один и тот же ФАУ можно пронумеровать разными логическими номерами.

Практически этот оператор используется для дисковых устройств, так как для них логический номер имеет дополнительное значение.

После выполнения оператора SELECT # 018 F, # 218 R, # 318 F по логическим номерам 0 и 3 можно обращаться к устройству 18 F (НГМД F), а по логическому номеру 2 — к устройству 18 R;

SELECT P [<цифра>] служит для задержки вывода информации. Время задержки определяет цифра. Единице соответствует 1/6 с, т. е. можно задать время задержки от 1/6 до 1,5 с. Используется оператор для задания паузы индикации. Операторы SELECT P и SELECT P0 отменяют паузу. SELECT P3 — пауза в 0,5 с.

SELECT $\begin{Bmatrix} D \\ R \\ G \end{Bmatrix}$ задает единицу измерения угловых величин в тригонометрических функциях (см. с. 57).

Оператор DEFFN. Оператор имеет следующую структуру:

DEFFN <имя FN> (<простая числовая переменная>) = <а.в.>
<**имя FN**> : := <цифра>| <буква латинского алфавита>.

Оператор DEFFN предназначен для определения функции пользователя FN.

Параметр <имя FN> используется для обращения к функции пользователя, определенной в программе оператором DEFFN.

Параметр <а.в.> — арифметическое выражение, определяющее функцию пользователя.

Параметр <простая числовая переменная> — формальный аргумент функции пользователя. Эта переменная может не входить в состав <а. в.>, т. е. может только обозначать условный аргумент.

При обращении к функции FN формальный аргумент следует заменять фактическим.

DEFFN — неисполняемый оператор, это просто обозначение функции, записанное в программе, а обращение к функции означает поиск этой записи по имени функции с целью вычислить значение функции как значение определяющего ее арифметического выражения. Фактическое значение формального аргумента, в свою очередь, задается арифметическим выражением.

Оператор DEFFN и обращение к функции могут занимать в программе произвольное положение относительно друг друга, т. е. не обязательно оператор записывать в строке с меньшим номером. Число обращений не ограничено. Оператор в непосредственном счете не используется.

Пример 19.

```
10 DEFFN:K(X)=X↑2-5*X:REM ОПРЕДЕЛЕНИЕ FNK(X)
20 PRINT FNK(7.5):REM ОБРАЩЕНИЕ
30 PRINT FNK(SQR(3.5)-5.7):REM ОБРАЩЕНИЕ
40 Y=7.5:PRINT FNK(Y):REM ОБРАЩЕНИЕ
```

Пример 20.

```
10 DEFFN M(A)=LOG(1.541)-X↑2
20 X=1:PRINT FNM(A)
30 REM ФОРМАЛЬНЫЙ ПАРАМЕТР НЕ ВХОДИТ В <A.B>
```

Оператор DEFFN′. Оператор DEFFN′ имеет две формы:

1) **DEFFN′** <номер СФ> <список символьных констант>
<список символьных констант> : := <символьная константа > [{; < символьная константа>} ...]

<номер СФ> — номер клавиши спецфункций (седьмой зоны), равный значению от 0 до 31 включительно.

2) **DEFFN′** <номер подпрограммы> (<список переменных>)
<номер подпрограммы> : := <целое>, принимающее значения в диапазоне 0+225
<список переменных> : := <переменная> [{, <переменная>}...]

Оператор DEFFN′ предназначен для:

ввода в память машины символьных констант, которые становятся значениями клавиш СФ;

организации помеченных подпрограмм.

Оператор формы 1 используется только в программе, но исполняется как оператор непосредственного счета сразу же после ввода и нажатия на клавишу CR/LF. Следовательно, уже при наборе следующей строки программы можно воспользоваться клавишей СФ для вызова

константы, закрепленной за клавишей оператором. Значение клавиши сохраняется, пока программа находится в памяти машины, и может быть использовано при наборе на клавиатуре вводимых данных в процессе счета по программе.

Оператор DEFFN′ позволяет использовать клавиши СФ как обычные информационные клавиши, значения которых задаются пользователем. В качестве значений обычно выбираются часто встречающиеся при наборе слова, фрагменты текста, которых нет на клавиатуре, например имена операторов матричных, операторов сортировки и т. д.

Пример 21.

В программе много раз встречается фрагмент «PRINT HEX (03): PRINT TAB (12);», его нужно закрепить за клавишей СФ, например третьей, и затем вызвать нажатием 3СФ.

```
10 DEFFN ′ 3"PRINT HEX(03):PRINT TAB(12);"
50 PRINT HEX(03):PRINT TAB(12);"СРЕДНИЕ"
110 PRINT HEX(03):PRINT TAB(22);"ОТКЛОНЕНИЯ"
```

При наборе строки 50 нажимается клавиша 3 СФ (седьмой зоны) и далее клавиши первой зоны для набора константы «СРЕДНИЕ»; аналогично следует поступать при наборе строки 110.

Оператор формы 2 используется только в программе для обозначения начала подпрограммы (см. с. 108). Список переменных в операторе означает список формальных параметров, фактические значения которых должны передаваться в подпрограмму при обращении к ней. Не допускается использование одного номера для разных подпрограмм.

Пример 22.

```
10 DEFFN ′ 5
70 DEFFN ′ 120(K,PX,M)
```

Глава VIII
ВВОД И ВЫВОД ДАННЫХ НА БОСГИ И ПУ

1. ОБЪЯВЛЕНИЕ ПЕРЕМЕННОЙ И МАССИВА, ПЕРЕОПРЕДЕЛЕНИЕ РАЗМЕРНОСТИ МАССИВА

Числа в машине всегда имеют постоянную разрядность и занимают постоянный объем памяти: целое — 2 байт, действительное — 8 байт.

Для символьных значений автоматически устанавливается длина в 16 символов, т. е. в памяти машины символьное значение занимает 16 байт. Допускается изменение длины символьного значения от 1 до 253 символов.

Размерность одномерных массивов — 10.

Размерность двумерных массивов — 10·10.

Допустимые значения размерности 1—7999 в пределах объема памяти. Объем памяти, занимаемой массивом, зависит от объема памяти, занимаемой значением элемента массива (см. с. 48).

Чтобы использовать в программе массивы с размерностью, отличной от стандартной, или символьные переменные, длина которых отлична от 16 байт, их нужно перед записью в программе объявить в операторе DIM. Оператор DIM устанавливает нужные пользователю размерность массивов и длину символьных переменных (простых и элементов масивов). В оперативной памяти для переменных и массивов резервируется место в соответствии с новыми параметрами.

Переменные и массивы, объявленные в DIM и необъявленные, относятся к категории необщих.

Когда программа состоит из двух и более последовательно загружаемых сегментов (частей), бывает необходимо передать значения некоторых переменных и массивов из одного сегмента в другой, сохраняя их в памяти. Эти переменные и массивы — общие для разных сегментов программы, т. е. относятся к категории общих.

Общие переменные независимо от типа, разрядности и размерности массивов обязательно нужно объявлять в операторе COM.

Переменные размещаются в памяти в такой последовательности, в какой они перечислены при объявлении в DIM и COM, а необъявленные переменные — в последовательности, в которой они встречаются в программе.

По требованию БЕЙСИК-системы общие данные должны размещаться в памяти раньше необщих, что обеспечивается порядком записи в программе операторов DIM и COM.

Переменные и массивы переменных можно переводить из категории общих в необщие и наоборот оператором COM CLEAR.

Изменить объявленную размерность массива и длину элемента символьного массива можно матричным оператором MAT REDIM, но это изменение должно быть в пределах объема памяти, занимаемой ранее объявленным массивом.

Оператор DIM. Оператор имеет следующую структуру:

DIM <список объявлений>
<**список объявлений**> : := <объявление> [{, <объявление>}...]
<**объявление**>: := <объявление символьной переменной> | < объявление массива>
<**объявление символьной переменной**> : := <простая символьная переменная> [<длина элемента>]
<**объявление массива**> (см с. 48).
<**длина элемента**> — целое число, задаваемое в диапазоне 1—253.

Оператор DIM предназначен для резервирования места в памяти для переменных и массивов, к которым будет обращение, а также для задания размерности одномерных и двумерных массивов, длины элементов символьных массивов, длины простых символьных переменных. В одном операторе DIM можно записать одно или несколько объявлений массивов и символьных переменных. Допускается также запись числовых переменных без указания параметра <длина элемента>,

так как изменить установленную в машине разрядность числа нельзя.

Оператор DIM должен находиться в программе перед первым обращением к объявляемым в операторе переменным, массивам и их элементам. Обращение к элементу массива в программе, необъявленному в DIM, означает автоматическое резервирование для массива стандартного объема памяти, как и при обращении к метке массива.

Элементы массива размещаются в памяти машины построчно элемент за элементом.

В программе может быть несколько операторов DIM.

Запись операторов имеет вид:

1) 10 DIM A(3) — объявлен действительный массив с тремя элементами A (1), A (2), A (3), которые располагаются в памяти последовательно в соответствии с номерами элементов;

2) 10 DIM B % (3,2), C (3), K (1,4), A (3,1) — объявлены три массива: целый двумерный массив, имеющий три строки и два столбца, действительный одномерный массив из трех элементов и два действительных двумерных массива (однострочный с четырьмя столбцами и одностолбцовый с тремя строками). Итак, объявлены переменные как совокупность всех элементов этих массивов:

B% (1,1) B% (1,2)	C(1)	K (1,1) K(1,2) K (1,3) K (1,4)	A (1,1)
B% (2,1) B% (2,2)	C(2)		A (2,1)
B% (3,1) B% (3,2)	C(3)		A (3,1)

Порядок следования элементов массива: B% (1,1); B% (1,2); B% (2,1), B% (2,2) и т. д.

3) 10DIMM ¤ (7), K ¤ (2)73, P ¤ (2,2) 1, R ¤ 115 — объявлены четыре символьных массива и символьная переменная R¤ с длиной 115 символов (байтов).

Массив с именем M¤ объявлен как одномерный с семью элементами стандартной длины.

Массив с именем K¤ — одномерный с двумя элементами, длина каждого элемента 73 символа, в массиве соответственно 146 символов.

Массив с именем P¤ — двумерный с двумя строками и двумя столбцами, т. е. содержащий четыре элемента длиной по 1 символу каждый.

Оператор COM. Оператор объявления общих переменных и массивов имеет следующую структуру:

COM <список объявлений>

<**список объявлений**> : := <объявление> [{, <объявление>...]

<**объявление**> : : = <объявление символьной переменной> | <объявление массива> | <простая числовая переменная>

Оператор COM предназначен для:

объявления простых переменных и массивов общими для нескольких программ (сегментов);

резервирования места в памяти для простых переменных и массивов (общих);

задания размерности массивов, длины простых символьных переменных и элементов символьных массивов.

Оператор COM должен исполняться в программе до использования любых переменных и массивов и перед первым оператором DIM.

Нельзя начинать счет программы по RUN <номер строки> с номера строки, в которой имеется оператор COM, если счет выполнялся хоть один раз по программе.

Пример 23.

```
10 COM K,M$
20 COM A,B%(2,4),C$(16,24)129,E#3
30 COM A(2,2),E#
100 COM A(2,2),E#
110 DIM C(3),L%(1,3)
120 M$="*1234567890*1234"
130 P(7)=16:REM ЭЛЕМЕНТ НЕОБЪЯВЛЕННОГО ДЕЙСТВИТЕЛЬНОГО
                МАССИВА СТАНДАРТНОЙ РАЗМЕРНОСТИ(10)
```

Оператор COM CLEAR. Оператор имеет следующую структуру:

COM CLEAR [{ ⟨простая числовая переменная⟩ / ⟨простая символьная переменная⟩ / ⟨метка массива⟩ }]

Оператор COM CLEAR предназначен для переопределения всех общих или нескольких последних общих в списке COM массивов и переменных в необщие, а также наоборот — всех необщих или нескольких необщих переменных и массивов в общие.

В общие переводятся необщие переменные и массивы с начала списка, т. е. с начала последовательности расположения их в памяти. Иначе говоря, оператор COM CLEAR перемещает условную границу между общими и необщими данными в памяти машины вперед (в зону необщих) или назад (в зону общих). Параметр COM CLEAR указывает место новой условной границы. Если параметр COM CLEAR был записан в списке COM, то все переменные и массивы, записанные в списке COM до него остаются необщими.

Если параметр COM CLEAR не был записан в списке COM, то все необщие переменные и массивы, встретившиеся в программе до него, становятся общими.

Если параметр опущен, все общие переменные и массивы становятся необщими. Значения переменных сохраняются.

Пример 24.

```
10 COM A,C,E(2,12)
20 DIM M(2):X=0.5
100 COM CLEAR C:REM ПОСЛЕ COM CLEAR ПЕРЕМЕННАЯ A ОСТАЛАСЬ ОБ
ЩЕЙ
```

Пример 25.

```
10 COM A(2,4),B(7)
20 C=16:K=24
30 COM CLEAR K:REM ПОСЛЕ COM CLEAR СТАНОВЯТСЯ ОБЩИМИ МАССИВЫ
 A,B И ПЕРЕМЕННАЯ C
```

Пример 26.

```
10 COM A,B,C%(2,75)
20 DIM K%6,M(299)
30 COM CLEAR
```

Все общие стали необщими.

Оператор MAT REDIM. Оператор имеет следующую структуру:

MAT REDIM <список>
<список> : := <объявление массива> [{, <объявление массива>}...]

Оператор предназначен для задания новой размерности (текущей) числовым и символьным массивам в пределах ранее объявленной и максимальной размерности массивов.

При переопределении общее количество элементов числовых массивов не должно увеличиваться по сравнению с исходными.

При переопределении символьных массивов учитывается вся цепочка символов массива независимо от разделения на элементы. Нельзя переопределять одномерный массив двумерным и двумерный одномерным. Один и тот же массив можно переопределять многократно.

Пример 27.

```
10 DIM A(5),B%(12)2,C%(4,2)3
20 MAT REDIM A(4),B%(1),C%(1,20)1,L(2,3)
```

В результате выполнения программы массивы, определенные в операторе DIM или по умолчанию (10·10), приобретают новую размерность.

Пример 28.

```
10 DIM A(5,5),B(5,5)
20 MAT INPUT A(2,2),B(2,2)
30 MAT REDIM A(5,2):MAT PRINT A
40 MAT REDIM B(2,5):MAT PRINT B
```

```
 1             2
 3             4
 0             0
 0             0
 0             0

 1             2             3             4             0
 0             0             0             0             0
```

При выполнении программы и ввода значений 1, 2, 3, 4, 1, 2, 3, 4 распечатывались значения массивов A и B после переопределения их размерности

Пример 29.

```
10 DIM A$(2,4)2
20 MAT INPUT A$
30 MAT PRINT A$
40 MAT REDIM A$(2,4)1
50 MAT PRINT A$
```

После ввода: AB, CD, EF, 12, 3X, GA, BL, KR первый оператор MAT PRINT выведет на экран дисплея данные матрицы A. Второй оператор MAT PRINT выведет на экран дисплея данные в виде матрицы

AB	CD	EF	12
3X	GA	BL	KR

A	B	C	D
E	F	1	2

При использовании оператора MAT REDIM значения элементов матрицы сохраняются, возможно только перераспределение индексов элементов в соответствии с заданными в операторе параметрами размерности матрицы.

2. ИСПОЛЬЗОВАНИЕ КОНСТАНТ В ПРОГРАММЕ

Исходные данные для обработки вводятся в машину разными способами. Константы вводятся и записываются в память машины вместе с программой в теле операторов. Например, тело оператора DATA полностью состоит из списка констант. Константы хранятся в памяти до тех пор, пока хранится в памяти программа.

Переменным значения присваиваются уже в процессе счета по программе. Значение каждой переменной может изменяться при счете многократно, причем новое значение переменной автоматически уничтожает прежнее.

Значением переменной может стать:

константа из оператора DATA, присвоенная оператором READ или MAT READ (если переменная — элемент массива);

константа, значение другой переменной, значение арифметического выражения, а также результат выполнения других операций, присвоенный переменной оператором LET и матричным оператором MAT=;

значение, присвоенное символьной переменной или элементам символьного массива специальным оператором INIT;

константы, присвоенные элементам числовых массивов матричными операторами MAT ZER, MAT CON, MAT IDN;

значение, введенное с клавиатуры или с внешнего устройства;

начальные значения (нули и совокупность пробела), присваиваемые в начале счета оператором RUN.

Оператор DATA. Оператор имеет следующую структуру:

DATA <элемент DATA> [{, <элемент DATA>}...]

<**элемент DATA**> : := $\left\{\begin{matrix}+\\-\end{matrix}\right\}$ <числовая константа> | <символьная константа>

Оператор **DATA** предназначен для записи и хранения констант. Оператор неисполняемый и может быть записан в любом месте программы. В непосредственном счете не используется.

В программе можно записывать несколько операторов DATA, при этом все константы всех операторов DATA программы считаются единой цепочкой со сквозной нумерацией. Порядок нумерации констант соответствует порядку записи операторов DATA.

Константы, перечисленные в DATA непосредственно, т. е. без записи их в переменные, в обработке участвовать не могут. Константы становятся значениями переменных после выполнения операторов READ и MAT READ.

Правильная синтаксическая запись оператора имеет вид:

```
10 DATA 2.5,-16,#PI,"ИТОГО"
20 DATA 17.247,"ABC",-1.5E-7,HEX(0741)
```

Оператор READ. Оператор имеет следующую структуру:

READ <переменная> [{,<переменная>}...]

Оператор READ предназначен для выборки констант из операторов DATA. При выполнении операторов цепочка констант из списка DATA, начиная с текущего номера, присваивается последовательно переменным списка READ. При этом обязательно соблюдение соответствия типа переменной типу присваиваемой константы. Номер текущей константы списка DATA устанавливается в 1 оператором RUN при запуске программы на счет, а после выполнения оператора READ номером текущей константы становится номер первой неиспользованной константы в общем списке DATA.

В программе могут использоваться несколько операторов READ. Каждый последующий исполняемый READ продолжает использовать константы из списка DATA. Если переменных в списке READ оказывается больше, чем констант в списке DATA к моменту исполнения оператора READ, то выдается сообщение об ошибке. Изменить номер текущей константы можно оператором RESTORE. Оператор MAT READ использует константы из того же списка DATA, изменяя текущий номер константы, как READ.

Пример 30.

```
5 REM-КОЛИЧЕСТВО КОНСТАНТ РАВНО КОЛИЧЕСТВУ ПЕРЕМЕННЫХ
10 DATA 2.4,16,"ABC",21
20 READ A,X%,E,C(6,4)
30 PRINT A,X%,E:PRINT C().
```

Пример 31.

```
5 REM-КОЛИЧЕСТВО КОНСТАНТ БОЛЬШЕ, ЧЕМ ПЕРЕМЕННЫХ
10 DATA 1.1,5,2.3,4.4,789
20 READ A,B:PRINT A,B
30 READ B,C,D:PRINT B,C,D,
```

Пример 32.

```
5 REM-ИСПОЛЬЗОВАНИЕ  НЕСКОЛЬКИХ ОПЕРАТОРОВ DATA
10 DATA 1.1,6,2.3
20 DATA 16,2.1,44,13.2
30 READ A,B:PRINT A,B
40 READ C,K,E:PRINT C,K,E
50 READ F,M:PRINT F,M
 1.1   6
 2.3   16   2.1
 44   13.2
```

Оператор RESTORE. Параметром оператора может быть арифметическое выражение:

RESTORE [<a.в.>]

Параметр <а. в.> принимает значения в диапазоне 1-255 в пределах фактического количества констант в списке операторов DATA. Дробная часть значения арифметического выражения не используется.

Оператор предназначен для задания в списке констант, указанных в операторах DATA текущего номера константы, которая будет использоваться операторами READ и MAT READ первой.

Если параметр <а. в.> не задан, то задается номер константы единица, т. е. действия операторов RESTORE и RESTORE 1 одинаковы.

Оператор RESTORE позволяет использовать константы из списка DATA многократно.

Пример 33.

```
10 DATA 1,2,3,4,5,6,7
20 READ A
30 RESTORE 5:READ B
40 RESTORE :READ C,D,E
50 PRINT A,B,C,D,E
```

Оператор LET. Оператор присвоения значения. Он имеет две формы, соответствующие типу используемых в операторе данных:

1) [**LET**] ⟨список числовых переменных⟩ = ⟨а. в.⟩

⟨**список числовых переменных**⟩ : : = ⟨числовая переменная⟩ [{, ⟨числовая переменная⟩}...]

2) [**LET**] ⟨список символьных переменных⟩ = { ⟨символьная переменная⟩ | ⟨символьная константа⟩ }

⟨**список символьных переменных**⟩ : : = ⟨символьная переменная⟩ [{, ⟨символьная переменная⟩}...]

Оператор LET предназначен для присвоения значения, находящегося справа от знака равенства, одной или нескольким переменным, указанным слева от знака равенства. При записи переменные разделяются символом «запятая». Имя оператора можно опустить.

Запись оператора имеет вид:

```
10 LET A=12.76
20 LET B,C,K=-648
30 A%,B%(7),C%=217
40 Y=2.5↑2-6.4*X+29.612
50 K$="СОРТ 1"
60 K1$=K$
70 A$,M$=HEX$(0A202A)
```

Если целой переменной присваивается значение, содержащее дробную часть, то дробная часть отбрасывается (без округления целой части).

Если символьной переменной присваивается значение длиннее (т. е. содержащее больше символов), чем символьная переменная, то последние (лишние) символы значения отбрасываются.

Если значение короче символьной переменной, то переменная дополняется символами «пробел» (HEX-код 20).

Пример 34.

```
10 DIM A$4,B$8
20 B$,A$="ИТОГО"
30 PRINT B$;A$;"***"
40 B,A%=2.645
50 PRINT B;A%;"***"


ИТОГО   ИТОГ***
 2.645  2 ***
```

STR-функция может быть записана в операторе LET как слева, так и справа от символа равно (=). Следует помнить, что расположенные справа присваиваемое значение и STR-функция означают выборку символов из символьной переменной или символьного массива, а расположенные слева переменная, которой присваивается значение, и STR-функция означают вставку символов значения в символьную переменную или символьный массив (аргумент STR-функции).

Пример 35.

```
10 A$="ПРОГРАММА"
20 B$=STR(A$,4,5):REM ВЫБОРКА ИЗ A$
30 PRINT B$
40 C$="КИЛОМЕТР"
50 STR(A$,4,3)=STR(C$,5,3):REM ВЫБОРКА ИЗ C$,ВСТАВКА В A$
60 STR(A$,7,3)="ЕЙ":REM ВСТАВКА В A$
70 PRINT A$


ГРАММ
ПРОМЕТЕЙ
```

Оператор часто используется и в программе, и в непосредственном счете.

Оператор INIT. Структура записи:

INIT (<аргумент>) <список INIT >

<**аргумент**> : := <hh> |'' <символ алфавита языка>''|
<символьная переменная>
<**список INIT**> : := <элемент INIT > [{, <элемент INIT > }...]
< **элемент INIT**> : := <символьная переменная> | < метка символьного массива>

Оператор INIT присваивает символьным переменным и символьным массивам, перечисленным в списке, значение, задаваемое параметром

аргумент. Если аргумент — символьная переменная, то оператор INIT использует ее первый символ. В результате выполнения оператора INIT каждый символ символьных переменных и символьных массивов списка становится равным символу, заданному параметром <аргумент>

Пример 36.

```
10 DIM A$12,B$3,C$1,K$(2,3)2
20 B$="ХОД"
30 INIT ("=")A$,K$():PRINT A$
40 INIT (21)C$:INIT (B$)B$
50 PRINT B$;C$;B$;C$;B$;C$
60 INIT (2D)A$:PRINT A$
70 PRINT K$()

============
XXX!XXX!XXX!
------------
======
======
```

3. ВВОД ДАННЫХ С КЛАВИАТУРЫ

Программируется ввод данных с клавиатуры операторами INPUT, LINPUT, MAT INPUT, KEYIN.

Клавиатура считается при этом консольным устройством ввода (CI — CONSOL INPUT).

Операторы INPUT и MAT INPUT используются для ввода чисел и символьных данных, LINPUT — для ввода и редактирования одного символьного данного, KEYIN — для ввода одного символа — HEX-кода нажатой клавиши (см. с. 101).

Клавиатура может быть использована как внешнее (периферийное) устройство ввода, тогда ввод данных программируется оператором ¤GI0.

Оператор INPUT. Оператор имеет две формы:

1) **INPUT** [<символьная константа>,] <переменная> [{, <переменная>} ...]

2) **INPUT** ¤ { <простая действительная> / <элемент действительного массива> }

Оператор предназначен для ввода с клавиатуры чисел и символьных данных в процессе выполнения программы, а также для ввода показания таймера.

Оператор INPUT (форма 1) содержит одну или несколько переменных любого типа, разделенных символом запятая, и символьную константу (необязательный параметр), которая при выполнении опера-

тора индицируется на экране. Символьная константа используется для указаний пользователю.

При выполнении оператора на экране индицируется символ «?», машина прекращает выполнение программы, т. е. переходит в режим ожидания набора данного. Если для консольного вывода CO (CONSOL OUTPUT)) заселектировано печатающее устройство, то и символьная константа и символ «?» не индицируются, а печатаются. Набираемое данное также печатается.

Если в операторе INPUT указана одна переменная, значит, нужно набрать одно данное того же типа, что и переменная, и нажать на клавишу CR/LF. Выполнение оператора заканчивается, и машина продолжает счет по программе далее. Если тип данного не совпадает с типом переменной, то выдаются сообщение об ошибке (ERR29) и символ «?», требующий повторения набора.

Если в операторе INPUT указано несколько переменных, то данные набираются последовательно в строгом соответствии с последовательностью записанных переменных. Завершается набор каждого данного (кроме последнего) нажатием клавиши с символом запятой или клавиши CR/LF; после набора последнего данного нажимается только CR/LF.

После запятой набираемое данное индицируется в той же логической строке, что и предыдущее. После нажатия на клавишу CR/LF происходит переход в начало следующей строки и индицируется символ «?», если было набрано не последнее данное. Таким образом, при наборе можно выбирать индикацию набора: все данные в одной строке, каждое данное в своей строке или группы данных (разбивка на группы произвольна) в строках. Ошибки можно исправлять только клавишей BACK SPACE в пределах одной набираемой группы данных (т. е. между символами «вопрос» (?) и «курсор»). Нажатие на клавишу CR/LF делает данное недоступным для редактирования.

Набор списка данных, вводимых в одном операторе, можно прервать двукратным нажатием на CR/LF. Переменные, для которых значения еще не введены, сохраняют прежние значения.

Число набирается по правилам записи числовой константы (нулевое значение набирать 0). Символьное данное набирается в кавычках, апострофах, HEX-функцией или просто цепочкой символов, которая не должна содержать символ «запятая», так как он является разделителем данных при наборе. Для набора используются все информационные клавиши, включая клавиши СФ, если им ранее были присвоены значения оператором DEFFN′ (см. с. 69).

Пример 37.

```
10 INPUT X,Y,Z$
20 INPUT "ДАННЫЕ",A,B
30 INPUT STR(A$,1,5)
40 INPUT K(1,2)
50 INPUT M$(3,4),B%(1,2)
```

```
10 DEFFN ' 1"ВСЕГО"
20 INPUT A%
30 PRINT A%
```

RUN CR/LF

? = При индикации символа «?» нажмите клавишу СФ 1

0 ВСЕГО — индикация результата набора

Пример 38.

```
10 DIM A(30)
20 N=1
30 INPUT "КОЛИЧЕСТВО",A(N)
40 N=N+1:GOTO 30
50 DEFFN ' 2
60 T=0
70 FOR B=1TO3
80 T=T+A(B)
90 NEXT B
100 PRINT "ИТОГО",T
110 N=1
120 RETURN
```

RUN : CR/LF

В результате выполнения программы на экране индицируется:

КОЛИЧЕСТВО
? 8
КОЛИЧЕСТВО
? 10
КОЛИЧЕСТВО
? 15
КОЛИЧЕСТВО
? — нажмите клавишу СФ 2
ИТОГО 33
КОЛИЧЕСТВО
?

Оператор INPUT (форма 2) присваивает переменной значение таймера, который воспринимается оператором как счетчик времени, работающий с дискретностью 1/2000 с. Счетчик убывающий, поэтому для определения интервала реального времени в секундах нужно пользоваться формулой:

(T1 — T2)/2000, где T1 —первое значение интервала времени, T2 — — последнее.

Пример 39.

```
20 INPUT #T1
30 DATA 2,4,6,8
40 READ A,B,C
50 PRINT A;C;B
60 INPUT #T2
70 PRINT "ВРЕМЯ СЧЕТА=";(T1-T2)/2000;"СЕК"
```

Оператор LINPUT. Оператор набора текста имеет следующую структуру:

LINPUT [<символьная константа>,] {<символьная переменная> / <метка символьного массива>}

Оператор LINPUT предназначен для ввода значения символьной переменной или символьного массива в режиме редактирования. Символьная константа — индицируемый комментарий (как в INPUT).

При выполнении оператора LINPUT на экране индицируется значение символьной константы или символьного массива (не более 240 символов), подлежащего редактированию. Курсор индицируется под первым символом значения. Строка редактируемого значения помечена символом «*».

Для набора и редактирования можно пользоваться: информационными клавишами, включая СФ; клавишами, управляющими перемещением курсора; клавишами редактирования (как в режиме EDIT при наборе программы). Ввод нового значения рассматривается как редактирование пустой строки.

Перемещать курсор можно только в пределах длины редактируемого значения. Завершается набор нажатием на клавишу CR/LF, после чего прежнее значение становится недоступным. До нажатия на CR/LF можно возвратить первоначальное значение редактируемой символьной переменной или символьного массива последовательным нажатием на клавиши LINE ERASE, EDIT, RECALL.

Оператор позволяет использовать для набора любые символы ограничители, т. е. кавычки, апострофы, запятую и пробелы.

Пример 40.

```
10 DIM A$*23
20 A$="ДОКУМЕНТ ВЫДАН   .  ."
30 LINPUT "НАБЕРИТЕ ЧЧ.ММ.ГГ",A$
40 PRINT A$

ДОКУМЕНТ ВЫДАН 12.10.85
```

Для набора даты подготовлена константа. Ввести дату в процессе счета.

4. ВЫВОД ДАННЫХ НА БОСГИ И ПЕЧАТАЮЩЕЕ УСТРОЙСТВО

Вывод информации в читаемом виде на экран БОСГИ и на печатающее устройство программируется специальными операторами печати PRINT, HEXPRINT, MAT PRINT, PRINTUSING. Можно также использовать БОСГИ и ПУ как любое нестандартное устройство (для которых в языке нет специальных операторов ввода-вывода) и программировать вывод оператором ¤ GIO.

Операторы печати позволяют вывести любую информацию по стандартным форматам печати и по форматам, задаваемым пользователем.

Оператор PRINT незаменим в непосредственном счете, а в программе он используется как универсальный оператор вывода, т. е. он не только выдает информацию в читаемом виде на экран и печать, но и управляет выводом и работой многих устройств ввода-вывода. Как оператор печати он наиболее удобен для вывода текста, т. е. данных в символьном виде, и для управления печатью, а также в качестве сервисного оператора.

Оператор PRINTUSING печатает (индицирует) информацию по формату, задаваемому пользователем, и является основным оператором формирования печатаемых таблиц.

Операторы MAT PRINT и HEXPRINT можно отнести к операторам сервисной печати.

HEXPRINT выводит значение символьной переменной HEX-кодами. Потребность в таком выводе незначительна, поэтому наиболее часто этот оператор используется при отладке и при обработке текстов.

MAT PRINT — оператор печати массива, удобен для вывода вспомогательной информации, для просмотра и анализа, но может быть использован и для печати таблиц, если они подготовлены и сформированы как значения элементов символьного массива.

Задание устройства вывода. Для всех операторов печати БЕЙСИК-системой предназначено БОСГИ (по умолчанию задания устройства). Задать устройство для вывода операторами печати можно в самом операторе параметром <устройство>:

⟨**устройство**⟩ : : = /⟨**ФАУ**⟩ | # ⟨логический номер⟩

Практически задание выполняется как «/<ФАУ>»: PRINT/05, PRINT/0C, HEXPRINT/0C.

Устройство с указанным ФАУ устанавливается с длиной строки, заданной ранее системой (80 символов) или оператором SELECT PRINT. Задание устройства в операторе распространяется только на выполнение данного оператора, а предыдущее задание устройства, выполненное системой или оператором SELECT PRINT, на это время отменяется.

Задать устройство печати сразу для всех операторов печати и задать новую длину строки можно оператором SELECT PRINT (см. с. 67).

Пример 41.

```
10 PRINT A:REM          -НА БОСГИ
20 PRINT /0C,"A":REM    -НА ПУ, ДЛИНА СТРОКИ 80
30 PRINT "A":REM        -НА БОСГИ
40 SELECT PRINT0C(130)
50 PRINT "B":REM        -НА ПУ, ДЛИНА СТРОКИ 130
60 PRINT /05,"B":REM    -НА БОСГИ
70 SELECT PRINT05(80)
80 PRINT "C":REM        -НА БОСГИ
```

Оператор PRINT. Оператор имеет две формы: PRINT и PRINT AT. Структура PRINT имеет вид:

PRINT[{ [⟨устройство⟩],] ⟨список PRINT⟩ / ⟨устройство⟩ }]

⟨**список** PRINT⟩: : = { { { ; / , } [⟨элемент PRINT⟩] } ... / ⟨элемент PRINT⟩ [{ { { ; / , } [⟨элемент PRINT⟩] } ...]] }

⟨**элемент** PRINT⟩: : = { ⟨а. в.⟩ / TAB (⟨а. в.⟩) / ⟨символьная переменная⟩ / ⟨символьная константа⟩ / ⟨метка массива⟩ }

Оператор PRINT предназначен для вывода на заданное в операторе устройство (а по умолчанию — на БОСГИ или устройство, заданное в SELECT PRINT) числа, символьного данного, значения массива, а также любой их совокупности.

Каждое выводимое значение, а также функция табуляции TAB (<а. в.>) задается в операторе параметром <элемент PRINT>.

Разделяются элементы символом «запятая» или «точка с запятой» ({ ; / , }). Эти разделители можно записывать до первого элемента и после последнего элемента, а также по два и более разделителей подряд в любом допустимом месте оператора, так как разделители управляют выводом, задавая стандартные форматы печати:

зонный формат, длина зоны 16 позиций; вывод выполняется с начала зоны, если перед элементом имеется символ «запятая»;

уплотненный или плотный формат; вывод выполняется с первой свободной позиции, если перед элементом имеется символ «точка с запятой».

Иначе можно сказать, что после вывода элемента курсор (печатающая головка) перемещается в начало первой свободной зоны, если после элемента стоит символ «запятая», и остается в первой свободной позиции, если после элемента стоит символ «точка с запятой». Запятая вызывает перемещение и в том случае, если элемента нет.

Оператор PRINT после вывода списка элементов выполняет перевод строки и возврат каретки, т. е. переход в первую позицию следующей строки, если в конце оператора нет разделителя. Наличие конечного разделителя в операторе PRINT запрещает перевод строки и возврат каретки, т. е. курсор (печатающая головка) остался в той позиции, в которой он оказался после вывода последнего элемента (в первой свободной позиции или в начале первой свободной зоны).

Пример 42.

```
10 PRINT :PRINT /0C:REM     -ПЕРЕВОД СТРОКИ И ВОЗВРАТ КАРЕТКИ
20 PRINT K,2.2,"ABC":REM    -ВЫВОД В ЗОННОМ ФОРМАТЕ
30 PRINT A;B%;"***":REM     -ВЫВОД В ПЛТНОМ ФОРМАТЕ
40 PRINT ,,7;"M",,"!":REM   -ВЫВОД 7 С 3-ЕЙ ЗОНЫ,! С 4-ОЙ ЗОНЫ,М-В
 3-ЕЙ ЗОНЕ
50 PRINT "H",,,25,:REM      -ВЫВОД Н С НАЧАЛА СТРОКИ,25 С 4-ОЙ ЗОН
Ы БЕЗ ПОСЛЕДУЮЩЕГО ПЕРЕВОДА СТРОКИ
60 PRINT "ИТОГ":REM         -ВЫВОД СЛОВА ИТОГ С НАЧАЛА 5-ОЙ ЗОНЫ
```

```
 0              2.2            ABC
 0  @ ***
                                7 M
                                                25              ИТОГ
```

Символьная константа печатается (индицируется) без ограничителей (кавычек и апострофов).

Символьная константа, заданная HEX-функцией, выводится символами, если HEX-коды являются кодами печатаемых символов; HEX-коды строчных букв выводятся соответствующими прописными буквами; непечатаемые коды либо выводятся пробелами, либо исполняются как управляющие (табл. 9.).

Пример 43.

```
10 PRINT /05,HEX(03):REM    -ОЧИСТКА ЭКРАНА
20 PRINT /05,HEX(07):REM    -ПОДАЧА ЗВУКОВОГО СИГНАЛА
30 PRINT HEX(0A0D):REM      -ПЕРЕВОД СТРОКИ И ВОЗВРАТ КАРЕТКИ
40 PRINT HEX(41420A3132):REM-ВЫВОД СИМВОЛОВ С ПЕРЕВОДОМ СТРОКИ
```

Значение символьной переменной выводится полностью вместе с конечными пробелами, т. е. число выводимых символов равно длине переменной. HEX-коды преобразуются так же, как при выводе константы.

Числовые константы выводятся в естественной форме, если их значение находится в пределах $0{,}1—10^{13}$, в противном случае — в экспоненциальной форме.

При выводе значения числовой переменной печатается знак числа: положительного — символом «пробел», отрицательного — символом «минус», значение элемента в естественной или экспоненциальной форме и пробел после него. Пробел — разделитель чисел на печати.

Функция табуляции TAB используется для задания номера печатной позиции (знакоместа) в строке, начиная с которого будет выводиться следующий элемент PRINT. Номером является значение арифметического выражения без дробной части и знака в диапазоне 0—255. Счет ведется с начала строки в пределах одной строки. При выполнении опе-

9. HEX-коды управления устройствами

Наименование команды по таблице КОИ-8	Код шестнадцатеричный	Реализация команды (процедуры)	
		БОСГИ	
		Символьный канал	Графический канал
НЗ	01	Установка курсора в начало экрана без очистки экрана	Установка курсора в начало координат экрана без очистки экрана
НТ	02	—	Перемещение курсора по диагонали ($-x$, $-y$)
КТ	03	Очистка экрана, установка в начало координат	Перемещение курсора по диагонали ($+x$, $-y$)
КП	04	—	Перемещение курсора по диагонали ($-x$, $+y$)
КТМ	05	Снятие блокировки курсора	Перемещение курсора ($+x$, $+y$)
ДА	06	Блокировка (гашение) курсора с сохранением его местоположения на экране	Установка режима «Засвет»
ЗВ	07	Подача звукового сигнала	Установка режима «Стирание засвета»
ВШ	08	Перемещение курсора на одно знакоместо влево	Перемещение курсора по горизонтали ($-x$)
ГТ	09	Перемещение курсора на одно знакоместо вправо	Перемещение курсора по горизонтали ($+x$)
ПС	0A	Перевод курсора вниз на ту же самую позицию	Перемещение курсора по вертикали ($-y$)
ВТ	0B	—	Установка режима «Сохранение информации»
ПФ	0C	Перевод курсора вверх на ту же самую позицию	Перемещение курсора по вертикали ($+y$)
ВК	0D	Установка курсора в начало строки	Очистка экрана, установка курсора, в начало координат
ВЫХ	0E	—	Блокировка вывода на экран графической информации
ВХ	0F	—	Снятие блокировки вывода на экран графической информации
СУ1	11	Установка режима индикации символов в негативе	
СУ2	12	Установка режима индикации символов в позитиве	
НС	85	Перевод курсора вниз в начало строки	—

Наименование команды по таблице КОИ-8	Код шестнадцатеричный	Реализация команды (процедуры)
		Графопостроитель (БИФ «Искра 015-33»)
НЗ	01	Установка пера в начало координат
НТ	02	Перемещение пера по диагонали ($-x$, $-y$)
КТ	03	Перемещение пера по диагонали ($+x$, $-y$)
КП	04	Перемещение пера по диагонали ($-x$, $+y$)
КТМ	05	Перемещение пера по диагонали ($+x$, $+y$)
ДА	06	Опускание пера
ЗВ	07	Поднятие пера
ВШ	08	Перемещение пера по горизонтали ($-x$)
ГТ	09	Перемещение пера по горизонтали ($+x$)
ПС	0A	Перемещение пера по вертикали ($-y$)
ВТ	0B	Поднятие пера
ПФ	0C	Перемещение пера по вертикали ($+y$)
ВК	0D	Установка пера в начало координат
		ДЗМ-180 (БИФ «Искра 015-33»)
ГТ	09	Табуляция печатающего блока в следующую позицию
ПС	0A	Возврат печатающего блока с переводом строки
ВТ	0B	Возврат печатающего блока с переводом на n строк, где n определяется программной перфолентой (1-я и 2-я дорожки)
ПФ	0C	Возврат печатающего блока с переводом в начало страницы, которое определяется по программной перфоленте
ВК	0D	Возврат печатающего блока с переводом строки
СУ2	12	Возврат печатающего блока с переводом на n строк, где n определяется по программной перфоленте
СУ3	13	Возврат печатающего блока с переводом в начало страницы, которое определяется по программной перфоленте
		РОБОТРОН-1154 (БИФ «Искра 015-30-01»)
ВШ	08	Возврат на шаг
ГТ	09	Табуляция в следующую позицию
ПС	0A	Перевод строки
ВК	0	Возврат каретки в начало строки
СУ1	11	Черный цвет ленты ПУ
СУ2	12	Красный цвет ленты ПУ

ратора PRINT курсор (печатающая головка) перемещается в позицию с указанным номером, если номер не меньше номера текущей позиции и не больше номера последнего символа строки по заданной длине. Если номер больше длины, то переход выполняется в первую позицию следующей строки. После функции TAB следует ставить ";".

Пример 44.

```
10 PRINT TAB(10);"ABC":REM      -ABC ЗАЙМЕТ ПОЗИЦИИ 10,11,12
20 X=215:PRINT TAB(5);X:REM     -ЧИСЛО 215 СО ЗНАКОМ ЗАЙМЕТ
                                ПОЗИЦИИ 5,6,7,8,9
30 PRINT "XXX";TAB(4);X;"X":REM-XXX  215 X

          ABC
      215
XXX  215 X
```

Если элементом PRINT является метка массива, то вывод значений элементов массива производится последовательно. Каждая строка массива выводится с новой строки экрана или ПУ, за исключением первой строки, когда метке массива предшествует элемент PRINT. Элемент PRINT, следующий за меткой массива, может выводиться в той же строке, что и последующий элемент массива, поэтому для вывода массива лучше записать отдельный оператор PRINT или воспользоваться оператором MAT PRINT.

Пример 45.

```
10 DIM A$(1,1)12:A$(1,1)="ПЕРЕМЕННАЯ"
20 PRINT A$()
30 MAT REDIM A$(3,4)1:PRINT A$():PRINT
40 MAT REDIM A$(2,2)3:PRINT A$():PRINT
50 PRINT "XXX";A$();"XXX"

ПЕРЕМЕННАЯ
ПЕРЕ
МЕНН
АЯ

ПЕРЕМЕ
ННАЯ

XXXПЕРЕМЕ
ННАЯ  XXX
```

Структура оператора PRINT AT имеет вид:

PRINT AT |<устройство>,| (<а.в.1>, <а.в.2> |, <а.в.3>|)

Оператор PRINT AT предназначен для установки курсора на экране БОСГИ в заданное положение и стирания указанного числа символов, начиная с заданного.

Параметр <а.в.1> указывает номер строки; параметр <а.в.2> — порядковый номер знакоместа в строке (1÷80), куда устанавливается курсор; параметр <а.в.3> — число стираемых символов на экране (если он не указан, то символы не стираются, а курсор устанавливается в указанное положение). Если значения параметров превышают размеры экрана, то курсор устанавливается по максимально допустимым параметрам.

После выполнения PRINT AT можно выводить на экран БОСГИ информацию операторами PRINT и INPUT. Оператор PRINT AT используется при программировании ввода, когда для удобства набора нужно разместить «подсказку» в заданном месте экрана.

В непосредственном счете оператор выполняется иначе, чем в программе. Оператор стирает заданное число символов, начиная с указанного; курсор не остается в этой позиции, а переходит в начало следующей строки, как при выполнении любого оператора в непосредственном счете.

Пример 46.

```
10 PRINT AT(7,5,16):REM -СТИРАЮТСЯ 16 СИМВОЛОВ В 7 СТРОКЕ,
                         НАЧИНАЯ С 5-ГО СИМВОЛА,КУРСОР
                         ОСТАЕТСЯ В 5 ПОЗИЦИИ
20 PRINT
25 PRINT AT(0,20,20)
```

Пример 47.

```
10 PRINT HEX(030A0A0A);
20 PRINT "1. ТЕМПЕРАТУРА-":PRINT
30 PRINT "2. ДАВЛЕНИЕ-":PRINT
40 PRINT "3. ВЛАЖНОСТЬ-":PRINT
50 PRINT "4. СКОРСТЬ ВЕТРА-":PRINT
60 INPUT "ЗАДАЙТЕ НОМЕР ВВОДИМОГО ПАРАМЕТРА",K
70 PRINT AT(12,1,70)
80 PRINT AT(2*K+2,19):INPUT M
90 PRINT AT(12,1):GOTO 60
```

Оператор HEXPRINT. Оператор имеет следующую структуру:

HEXPRINT [⟨устройство⟩,] ⟨элемент HEXPRINT⟩ $\left[\left\{\left\{\begin{matrix},\\;\end{matrix}\right\}\right.\right.$ [⟨элемент HEXPRINT⟩]}...]

⟨**элемент** HEXPRINT⟩ : : = $\left\{\begin{matrix}\text{⟨символьная переменная⟩}\\ \text{⟨метка символьного массива⟩}\\ \text{TAB (⟨а. в.⟩)}\end{matrix}\right\}$

Оператор HEXPRINT предназначен для индикации (печати) символьной информации шестнадцатеричными кодами (HEX-кодами) (см. табл. 7).

В операторе HEXPRINT может быть записан один или несколько элементов HEXPRINT, разделенных символом «запятая» или «точка с запятой».

При выполнении оператора каждый символ значения символьной переменной или символьного массива, включая конечные пробелы, выводится в виде двухразрядного шестнадцатеричного числа (HEX-кодом) в соответствии с КОИ-8 (см. табл. 7).

Разделитель «точка с запятой» означает вывод следующего элемента HEXPRINT с первой свободной позиции в той же строке, а запятая означает вывод следующего элемента с начала следующей строки, т. е. запятая выполняет перевод строки и возврат в начало ее.

Значения в строке печатаются всегда плотно с автоматическим переходом в следующую строку для вывода не поместившихся в текущей строке символов. Функция TAB выполняется так же, как в PRINT (см. с. 86).

Пример 48.

```
10 DIM A$*5,B$*5,C$*5,M$(1,4)*3
20 A$="ABC":B$="A+B+C":C$="A BC"
30 HEXPRINT A$,B$;C$
40 HEXPRINT A$;B$;C$
50 INIT ("X")M$():HEXPRINT M$()

4142432020
412B4228434120424320
4142432020412B4228434120424320
5858585858585858585858
```

Оператор PRINTUSING. Оператор печати по формату имеет следующую структуру:

PRINTUSING [⟨устройство⟩,] ⟨формат PU⟩ [, ⟨элемент PRINT⟩

$$\left[\left\{\left\{\begin{matrix};\\ ,\end{matrix}\right\}\text{⟨элемент PRINT⟩}\right\}\ldots\right]$$

$$\text{⟨формат PU⟩}::=\left\{\begin{array}{l}\text{⟨символьная переменная⟩}\\ \text{„⟨цепочка символов⟩“}\\ \text{⟨номер строки с оператором \% (IMAGE)⟩}\end{array}\right\}$$

Значение параметра <элемент PRINT> (см. с. 85).

Оператор PRINTUSING предназначен для вывода числовой и символьной информации на устройство в заданном пользователем формате.

В операторе PRINTUSING обязательным параметром является формат PU. Он является форматом вывода, выполняемого оператором, и

описывает вид одной строки выводимой таблицы, части выводимой строки или нескольких строк.

Формат PU задается следующим образом:

цепочкой символов в кавычках;

символьной переменной, которой предварительно присвоено значение цепочки символов; длина символьной переменной объявляется оператором DIM по длине формата PU с учетом конечных пробелов значения символьной переменной; конечные пробелы выводятся как символы комментария;

номером строки, где записан символ «%» и цепочка символов; символ «%» — имя оператора IMAGE; оператор % — неисполняемый и может быть записан в любом месте программы, но всегда с начала строки, а цепочка символов не должна содержать символ «двоеточие».

Задание формата символьной переменной или оператором % дает возможность использовать его в нескольких операторах PRINTUSING.

Приведем примеры задания формата:

```
10 PRINTUSING "#.# +#.### ";X;SIN(X)
110 DIM F$11:F$="#.# +#.### "
120 PRINTUSING F$;X;COS(X)
210 PRINTUSING 220;X;SIN(X)
220 %#.# +#.###
230 PRINTUSING 220;4;COS(4)
```

Формат PU может состоять из формата одного данного или форматов нескольких данных, а также из символов комментария. Символы комментария должны находиться между форматами данных, при этом они являются разделителем форматов данных. Символы комментария могут предшествовать первому формату данного и следовать за последним форматом данного. Формат PU может содержать только символы комментария, но обязательно должен содержать хотя бы один символ.

Комментарий (разделитель форматов данных) содержит чаще всего один или несколько символов «пробел», а также любую цепочку символов (букв, цифр, знаков), отличных от символов, используемых в формате данного (+, —, ,,.., #, ^, ¤, ;).

При выполнении оператора PRINTUSING выводимое данное редактируется по формату данного; символы комментария выводятся полностью строго в той последовательности, в какой записаны в формате PU.

Формат данного и правила редактирования данного. Формат данного — числовые и символьный форматы. Он описывает форму отредактированного выводимого данного соответствующего типа и назначение каждой позиции строки вывода, занятой выводимым данным.

Символьный формат — формат символьного значения. Он содержит символы «#», число которых равно числу печатных позиций (знакомест) строки вывода, занимаемых символами значения.

Формат ##### означает, что символьное значение будет занимать в строке пять позиций.

Если выводимое символьное значение короче формата, оно дополняется справа до формата незначащими символами «пробел».

Если символьное значение длиннее формата, то выводится часть значения с первого символа по числу # в формате.

К числовым форматам относятся формат 1 и формат 2.

Формат 1 задает вывод числа в естественной форме.

Для целого числа формат 1 — это цепочка символов, количество которых равно разрядности выводимого числа. Знаковая позиция в формате обозначается символами плюс, минус или не указывается. Например: ###, +###, −###.

Для вывода числа с дробной частью дополнительно используется символ «точка», отделяющий целую часть от дробной и в формате 1, и при выводе числа (#.##, +#.#, −##.#).

Если целое (целая часть) по разрядности меньше числа позиций целого (целой части) формата, то оно дополняется до формата слева незначащими пробелами, а если больше — выдается формат 1. Если дробная часть числа по разрядности меньше числа позиций дробной части формата, то она дополняется до формата справа незначащими нулями, если больше — не поместившиеся справа разряды отбрасываются без округления. При выводе действительного числа по формату целого отбрасывается вся дробная часть (табл. 10).

Знак отрицательного числа печатается символом «минус» независимо от того, указана знаковая позиция в формате или нет. Если не указана, то для знака используется дополнительная позиция (сверх формата PU), что вызывает смещение вправо всех последующих символов строки вывода.

Знак положительного числа выводится символом «плюс», если в формате 1 указан плюс; символом «пробел», если в «формате 1» указан минус; не выводится, если знаковая позиция в формате 1 не указана.

Формат 2 задает вывод числа в экспоненциальной форме с нормализованной или ненормализованной мантиссой. Позиции порядка в формате 2 указываются символами ^^^^, мантисса — как в формате 1: #. #^^^^, +.##^^^^, −#^^^^

10. Примеры редактирования чисел оператором PRINTUSING

Число	Форматы						
	###	+##	−###	−#.###	##.##	+#####	−#.##^^^^
25	25	+25	25	−#.####	2 5.0 0	+25.00	2.50E+01
−125	−125	+##	−125	−#.####	##.##		−1.25E+02
224	224	+##	224	##.####	##.##		2.24E+02
16.5	16	+16	16	−#.####	1 6.5 0	+16.50	1.65E+01
−2.3	− 2	−2	−2	−2.300	− 2.3 0	−2.30	−2.30E+00
8.6	8	+8	8	8.600	8.6 0	+8.60	8.60E+00
−1.556	− 1	−1	−1	−1.556	− 1.5 5	−1.55	−1.55E+00
675.5	675	+##	675	−#.####	##.##		6.75E+02
0.003	0	+0	0	0.003	0.0 0	+0.00	3.00E−03

Выводимое число редактируется по формату, и если показатель степени по модулю больше 99, то выдается формат. Правила вывода знака те же, что в формате 1.

В числовой формат может быть включен символ ¤ (денежный знак), который выводится между знаком числа и старшим разрядом числа — ¤ 2.47.

Если в формате данного имеется только комментарий, то PRINTUSING выводит только комментарий.

Соотношение между числом форматов данных в формате PU и числом элементов PRINT, перечисленных в операторе, может быть различным (равно, меньше, больше), но всегда формат PU используется для вывода всего списка элементов PRINT. В зависимости от этого соотношения разделители между элементами PRINT могут быть управляющими (запятая выполняет переход в начало следующей строки, а точка с запятой не меняет положения курсора и печатающей головки) или неуправляющими. Возможны случаи:

1) формат PU содержит один формат данного, а в списке несколько элементов PRINT; тогда все элементы выводятся по одному формату данного и все разделители между ними являются управляющими;

2) формат PU содержит несколько (K) форматов данных, но меньше, чем число элементов PRINT в списке; тогда каждые K элементов будут выводиться по формату PU и разделитель за каждым K-м элементом будет управляющим;

3) формат PU содержит форматов данных не меньше, чем число элементов PRINT в списке; тогда каждый элемент выводится по своему формату данного и все разделители неуправляющие.

Пример 49.

```
10 DIM C$5:C$="-#.##":A=-2:C=1.761
20 PRINTUSING C$,A;C,A,C;
30 PRINTUSING C$,A,C

-2.00 1.76
-2.00
 1.76-2.00
 1.76
```

Пример 50.

```
10 A=1:B=2:C=3
20 PRINTUSING "X=# F(X)=## ",A,FN1(A),B;FN1(B);C,FN1(C)
30 DEFFN 1(X)=X↑2+A

X=1 F(X)= 2
X=2 F(X)= 5 X=3 F(X)=10
```

Пример 51.

```
10 A$="СТОЛ":B%=10:C=29.7
20 PRINTUSING 30;A$,B%;C,B%*C
30 %###### ## ##.## ####.##

СТОЛ  10 29.70  297.00
```

Пример 52.

```
10 DIM A$*5,B$*5:A$="ИТОГО":B$="12"
20 PRINTUSING "####";B$;A$;"***"

12  ИТОГ***
```

Глава IX
УПРАВЛЕНИЕ ПРОЦЕССОМ СЧЕТА

1. ОБЩИЕ СВЕДЕНИЯ

В любой программе при упрощенном рассмотрении ее функциональной структуры можно выделить следующие функциональные блоки:

ввода исходных данных;

обработки;

вывода результатов на печать и другие устройства вывода.

Под блоками понимается вся совокупность операторов, реализующих, например, ввод исходных данных.

Программа, обеспечивающая решение задачи лишь в исключительных случаях бывает линейной, состоящей лишь из операторов ввода, печати, преобразований и т. д., которые выполняются один раз в строгой последовательности, соответствующей последовательности записи их в программе.

Обработка может включать в себя элементы анализа, проверки условий, от выполнения которых зависит выбор дальнейшего пути и способа обработки. Да и процесс ввода (вывода) часто требует определенной организации. Поэтому операторы организации управления счетом являются в программе определяющими, от них зависят качество и скорость вычислений, эффективность использования оперативной памяти, удобства эксплуатации программы в дальнейшем, наиболее четкая и полная реализация алгоритма.

Например, необходимо запрограммировать ввод нескольких (К) чисел по одному для последующей обработки их в машине. Решить такую задачу можно реализацией одного из алгоритмов:

ввести первое число, ввести второе число и т. д., ввести последнее (К-е) число;

ввести число, перейти к выполнению предыдущей команды; так можно вводить любое количество чисел, в частности К чисел:

ввести число; если введенное число не последнее, то перейти к дальнейшей обработке. (Следует уточнить понятие «последнее число», т. е. указать конкретно критерий оценки конца ввода. «Последнее число» — это может быть К-е число по счету или число, имеющее конкретное значение, т. е. число—признак конца ввода, тогда оно должно быть дополнительным к вводимым, т. е. (К + 1-м по счету);

выполнить К раз команду ВВЕСТИ ЧИСЛО.

Первый алгоритм прост, он реализуется программой, состоящей из К операторов ввода данных с клавиатуры (INPUT или LINPUT), но может быть использован лишь при малых значениях К. Для реализации второго алгоритма нужно использовать два оператора: оператор ввода и оператор, соответствующий команде ПЕРЕЙТИ К. Этот алгоритм также прост, но он не обеспечивает «устойчивого» решения, когда нужно вводить строго К чисел, так как машина не будет контролировать количество введенных чисел, а человеку свойственно ошибаться. Для реализации третьего алгоритма (точнее двух близких по реализации алгоритмов) кроме оператора ввода используется оператор, анализирующий выполнение условия и выполняющий переход, т. е. оператор, соответствующий команде ЕСЛИ ... ТО ПЕРЕЙТИ К. Для реализации последнего алгоритма используются операторы цикла и оператор ввода.

К операторам управления счетом можно отнести операторы программных переходов, организации циклов и подпрограмм.

2. ПРОГРАММНЫЕ ПЕРЕХОДЫ

Естественную последовательность выполнения операторов в программе изменяют многие операторы. Среди них операторы цикла FOR, NEXT; операторы обращения к подпрограммам и возврата из них GOSUB, GOSUB; RETURN.

Основную группу составляют операторы программных переходов:

GOTO — оператор безусловного перехода;

IF THEN — оператор условного перехода;

ON GOTO, ON GOSUB — операторы вычисляемого программного перехода;

ON ERROR — оператор перехода по сбою;

IF END THEN — оператор, выполняющий переход, если при вводе информации из файла с технического носителя встретился закрывающий блок;

KEYIN — оператор опроса состояния клавиатуры и прерывания счета с клавиатуры с вводом значения и выполнением последующего программного перехода.

Операторы программного перехода в качестве метки перехода используют номер строки, поэтому оператор, к выполнению которого нужно перейти, должен стоять в строке первым.

Оператор GOTO. Оператор программного перехода имеет следующую структуру:

GOTO ⟨номер строки⟩

Оператор предназначен для выполнения безусловного программного перехода для продолжения счета по программе с первого оператора строки, номер которой указан в операторе. Переход выполняется всегда, следовательно, в строке оператор GOTO должен быть последним исполняемым оператором, далее может быть записан только оператор REM.

В непосредственном счете оператор GOTO используется лишь в сочетании с HALT/STEP для задания отладочного режима счета.

Пример 53.

```
10 INPUT X
20 PRINT "X=";X,"SIN(X)=";SIN(X)
30 GOTO 10:REM ВОЗВРАТ К ОПЕРАТОРУ СТРОКИ 10
```

Пример 54.

```
10 INPUT X
20 IF X<0THEN30:GOTO 40:REM ПРОПУСК СТРОКИ 30,ЕСЛИ X>ИЛИ=0
30 X=1-X
40 PRINT X:GOTO 10
```

Оператор IF THEN. Оператор условного перехода имеет две формы, соответствующие типу сравниваемых данных:

1) **IF** { ⟨символьная переменная⟩ / ⟨символьная константа⟩ } ⟨операция сравнения⟩ { ⟨символьная переменная⟩ / ⟨символьная константа⟩ } THEN ⟨номер строки⟩

2) **IF** ⟨а. в.⟩ ⟨операция сравнения⟩ ⟨а. в.⟩ THEN ⟨номер строки⟩

⟨**операция сравнения**⟩: : = ⟨|⟨ = | = | ⟩ | ⟩ = | ⟨ ⟩

Оператор предназначен для выполнения программного перехода к первому оператору в строке с указанным номером, если сравниваемые операнды находятся в указанном отношении. Если отношение операндов отлично от указанного, после оператора IF THEN выполняется следующий за ним в программе оператор. Иначе можно сказать, что при выполнении оператора программный переход происходит по номеру строки, если код условия «ДА», и к следующему оператору, если код условия «НЕТ».

Сравниваемые операнды должны быть только одного типа. Символьные операнды сравниваются с учетом конечных пробелов. Если один операнд короче, то для сравнения он дополняется справа пробелами HEX (20).

Оператор условного перехода используется в программе для организации ветвящихся алгоритмов; в непосредственном счете он не используется.

Пример 55.

```
10 DIM A$*4:A$=HEX$(41424317)
20 IF A$<"ABC"THEN30:PRINT "ABC-МЕНЬШЕ":END
30 PRINT "HEX$(41424317)-МЕНЬШЕ"
```

Пример 56.

```
10 INPUT X
20 IF X<=7.548THEN40
30 F=2*X+9.17:PRINT X,F:GOTO 10
40 PRINT " КОНТРОЛЬНОЕ ЗНАЧЕНИЕ"
```

Оператор ON. Структура оператора:

$$\textbf{ON}\ <\text{а.в.}>\ \begin{Bmatrix}\text{GOTO}\\ \text{GOSUB}\end{Bmatrix}<\text{номер строки}>\ [\{,\ <\text{номер строки}>\}\ldots]$$

Оператор ON предназначен для выполнения вычисляемого перехода в программе.

Если в операторе указан параметр GOTO, то выполняется переход к строке с номером из списка номеров в операторе. Если в операторе указан параметр GOSUB, то выполняется обращение к подпрограмме, начинающейся со строки с номером из списка номеров.

Выбор номера строки перехода из списка задается значением арифметического выражения: значение арифметического выражения — порядковый номер строки перехода в списке. Если значение арифметического выражения меньше 0,5 и больше количества номеров списка, то оператор ON выполняет переход к следующему оператору.

Оператор используется, например, для организации блока программы ДИСПЕТЧЕР и обращения к различным участкам программы, реализующим различные технологические режимы. В непосредственном счете оператор не используется.

Пример 57.

```
5 PRINT CHR$(05);"ЗАДАВАЙТЕ ПОСЛЕДОВАТЕЛЬНО  ЧИСЛА 1-9"
10 INPUT X
20 ON (X+.5)/2GOTO40,60,50
30 PRINT "ЗАДАН НЕДОПУСТИМЫЙ ПАРАМЕТР":GOTO 10
40 PRINT "ВЫПОЛНЕН ПЕРВЫЙ ПЕРЕХОД":GOTO 10
50 PRINT "ВЫПОЛНЕН ПОСЛЕДНИЙ ПЕРЕХОД":GOTO 10
60 PRINT "ВЫПОЛНЕН ВТОРОЙ ПЕРЕХОД":GOTO 10
```

Пример 58.

```
20 PRINT "ЗАДАЙТЕ РЕЖИМ СЧЕТА:"
26 PRINT "ВВОД СПРАВОЧНИКА-1"
30 PRINT "РЕДАКТИРОВАНИЕ СПРАВОЧНИКА-2"
40 PRINT "РАСПЕЧАТКА СПРАВОЧНИКА-3"
50 INPUT A
60 ON AGOTO30,140,220
70 PRINT "ПОВТОРИТЕ ЗАДАНИЕ РЕЖИМА":GOTO 50
80 REM ВВОД СПАВОЧНИКА
130 STOP "КОНЕЦ ВВОДА СПРАВОЧНИКА"
140 REM РЕДАКТИРОВАНИЕ СПРАВОЧНИКА
210 STOP "КОНЕЦ РЕДАКТИРОВАНИЯ СПРАВОЧНИКА"
220 REM РАСПЕЧАТКА СПРАВОЧНИКА
250 STOP "КОНЕЦ РАСПЕЧАТКИ СПРАВОЧНИКА"
```

Оператор IF END THEN. Оператор программного перехода имеет параметром номер строки:

IF END THEN <номер строки>

Оператор IF END THEN предназначен для определения состояния файла данных при считывании информации из него.

Если при считывании встретился закрывающий блок файла (конечный), то оператор IF END THEN выполняет переход к оператору, первому в строке с указанным номером; если не встретился, то выполняет переход к следующему оператору программы. Оператор IF END THEN записывается в программе после операторов DATA LOAD DC, DATA LOAD DA, DATA LOAD BA

Пример 59.

```
10 DATA LOAD DC A,B,C()
20 IF ENDTHEN 40
30 PRINT A,B:PRINT C():GOTO 10
40 PRINT "КОНЕЦ ФАЙЛА"
```

Оператор ON ERROR. Оператор программного перехода по ошибке имеет следующую структуру:

ON ERROR <символьная переменная>, <символьная переменная> GOTO <номер строки>

Оператор ON ERROR предназначен для программной обработки сбойной ситуации, возникшей в процессе счета по программе.

Если оператор ON ERROR не используется, все сбойные ситуации (ошибки), возникающие в процессе счета, обрабатываются системой (см. с. 36). При этом анализируется причина сбоя, индицируется код ошибки и происходит останов. Выполнение оператора ON ERROR с параметрами вызывает отказ от системной обработки сбоя и переход к

программной обработке. Это значит, что возникший после выполнения ON ERROR любой сбой вызовет прерывание счета и программный переход к строке с номером, указанным в операторе. Эта строка является началом блока программы, обрабатывающего сбой.

Сведения об ошибке (которые при системной обработке индицируются на консольном устройстве вывода) заносятся автоматически при сбое в символьные переменные, указанные в операторе: в первую переменную код ошибки, во вторую — номер строки, в которой обнаружена ошибка.

Итак, если в программе используется оператор ON ERROR с параметрами, то в программе обязательно должен быть и блок обработки ошибки. Если ошибку можно устранить программно, то в блоке записываются операторы, устраняющие ошибку, и оператор возврата в основную программу для продолжения счета. Нельзя возвращаться в цикл и в подпрограмму.

Действие оператора ON ERROR отменяется выполнением другого ON ERROR с параметрами. Отказ от программной обработки ошибки и переход к системной обработке происходит при выполнении ON ERROR без параметров.

Если в процессе счета сбоя не было, то блок программной обработки сбоя остается неиспользованным, т. е. перехода к нему не происходит.

Если программа состоит из самостоятельных последовательно вводимых в память машины сегментов, то символьные переменные должны быть объявлены в операторе COM общими. Длина символьных переменных составляет 4 байт.

Пример 60.

```
5 DIM A%4,B%4
10 ON ERROR A%,B%GOTO200
70 ON ERROR
200 REM ПРОГРАММА ОБРАБОТКИ СБОЕВ
```

Пример 61.

```
10 DIM A%4,B%4
20 ON ERROR A%,B%GOTO60
30 READ X:PRINT X,
40 GOTO 30
50 DATA 1,2,3,4,5,11,12,21,"A"
60 ON ERROR :PRINT :PRINT "СПИСОК КОНСТАНТ ИСЧЕРПАН"
70 PRINT A%,B%

 1  2  3  4  5  11  12  21
СПИСОК КОНСТАНТ ИСЧЕРПАН
43              0030
```

В данном примере оператор ON ERROR используется для того, чтобы выбрать константы из оператора DATA без задания их числа. Присвоение символьной константы «А» из DATA действительной переменной X вызовет ошибку 43, счет прервется, и выполнится переход к строке 60, в которой без дополнительного анализа ошибки печатаются сообщение и значения символьных переменных с кодом ошибки и номером строки сбоя. Программная обработка далее отменяется.

Оператор KEYIN. Структура оператора:

KEYIN <символьная переменная>, <номер строки 1>, <номер строки 2>

Оператор KEYIN предназначен для опроса состояния консольного устройства ввода. По умолчанию ввод выполняется с клавиатуры с ФАУ 01. Изменить устройство ввода для оператора KEYIN можно оператором SELECT CI.

Если к моменту выполнения оператора KEYIN во входной регистр поступила информация (HEX-код нажатой клавиши), то происходит переход к строке с одним из указанных номеров, а информация (1 байт) записывается в символьную переменную, указанную в операторе. Если была нажата клавиша СФ (любая из седьмой зоны), то переход выполняется по номеру строки 2, в противном случае — по номеру строки 1.

Если к моменту выполнения оператора KEYIN информация не поступила (клавиша нажата не была), то выполняется переход к следующему оператору. Этот оператор напоминает выполнение двух операторов: ON GOTO и INPUT, но по INPUT счет останавливается и машина ждет ввода, а по KEYIN ожидания нет. Ожидание можно выполнить программное.

Пример 62. Для ввода символа, задающего режим дальнейшего счета, можно воспользоваться программой:

```
5 DIM A$*1
10 PRINT "ДЛЯ РЕДАКТИРОВАНИЯ НАЖМИТЕ ЛЮБУЮ КЛАВИШУ СФ"
20 PRINT "ДЛЯ ВЫПОЛНЕНИЯ СЧЕТА-ЛЮБУЮ ДРУГУЮ КЛАВИШУ"
30 KEYIN A$,40,100:GOTO 30
40 REM НАЧАЛО СЧЕТА
50 REM НАЧАЛО СЧЕТА
90 STOP "КОНЕЦ СЧЕТА"
100 REM РЕДАКТРОВАНИЕ
200 STOP "КОНЕЦ РЕДАКТИРОВАНИЯ"
```

Примечание. Клавиша CR/LF оператором KEYIN воспринимается как любая информационная (не СФ) с HEX (5С) (см. табл. 2).

3. ОРГАНИЗАЦИЯ ЦИКЛОВ

Программную реализацию циклического процесса принято называть циклом. Иначе цикл можно определить как совокупность операторов (записанных подряд), выполнение которых многократно повторяется, повторения следуют одно за другим.

Циклические процессы очень распространены, и в каждом языке есть специальные операторы, предназначенные для организации цикла. В языке БЕЙСИК операторы цикла — FOR, NEXT.

Кроме специальных операторов цикла для организации цикла могут быть использованы оператор GOTO (если цикл повторяется бесконечно), а также сочетание оператора GOTO с одним из операторов IF THEN, IF END THEN, ON ERROR, KEYIN.

Рассмотрим пример последовательного ввода K чисел с клавиатуры. Несколько алгоритмов решения приведены выше.

Пример 63.

```
10 REM БЕСКОНЕЧНЫЙ ЦИКЛ
20 INPUT X:PRINT X
30 GOTO 20
```

После запуска на счет и ввода необходимого количества чисел в ожидании очередного набора числа (индикации? —) можно прервать счет клавишей RESET.

Пример 64.

```
10 REM ВВОД ПЯТИ ЧИСЕЛ
20 M=M+1:REM M-СЧЕТЧИК ВВОДИМЫХ ЧИСЕЛ
30 INPUT X:PRINT X
40 IF M<5THEN20:END
```

Пример 65.

```
10 REM ВВОД ЧИСЕЛ ДО ПРИЗНАКА КОНЦА ВВОДА
20 INPUT "ВВЕДИТЕ ОЧЕРЕДНОЕ ЧИСЛО ИЛИ 9999-ПРИЗНАК КОНЦА";X
30 PRINT X
40 IF X<>9999THEN20:END
```

Пример 66.

```
10 REM ОПЕРАТОР ЦИКЛА.ВОД ПЯТИ ЧИСЕЛ
20 FOR K=1TO5:REM НАЧАЛО ЦИКЛА
30 INPUT X:PRINT X
40 NEXT K:REM КОНЕЦ ЦИКЛА
```

Операторы FOR, NEXT. Операторы FOR и NEXT при организации цикла всегда используются вместе. FOR — заголовок цикла, он записывается первым в группе операторов цикла, а NEXT последним. В программе может быть много циклов, и операторы FOR, NEXT, относящиеся к одному циклу, имеют одну и ту же переменную — переменную цикла.

Структура оператора **FOR**:

FOR <простая числовая переменная> = <а.в.1> **TO** <а.в.2> [**STEP** <а.в.3>]

Параметр <простая числовая переменная> — переменная цикла.

Параметр <а.в.1> задает ее начальное значение. Оно присваивается переменной цикла при выполнении оператора FOR. Параметр <а.в. 2> определяет конечное значение переменной цикла, которое она имеет при естественном выходе из цикла. Значение <а. в. 2> не всегда равно конечному значению. Оно может отличаться на значение, меньшее шага (т. е. значения <а.в.3>). Шаг изменения переменной цикла может быть в операторе не указан, тогда он равен единице.

Значение переменной цикла при каждом проходе цикла изменяется на шаг. Если шаг отрицательный, то значение уменьшается.

Итак, можно сказать, что FOR задает число повторов выполнения операторов, составляющих тело цикла, т. е. выполняющихся от FOR до NEXT. Обозначив соответствующие арифметические выражения буквой К с номером арифметического выражения, получим:

число повторов K = INT (ABS ((K2 — K1)/K3)) + 1

Пример записи заголовка цикла.

1) FOR A = 1 TO 5 — цикл на 5 повторов. A — переменная цикла. Значение A изменяется от 1 до 5 включительно с шагом 1, т. е. A принимает значения 1, 2, 3,4, 5

2) FOR K % = 0 TO 7 STEP2 — цикл выполняется 4 раза. Значение K % изменяется от 0 до 6 с шагом 2, т. е. K % принимает значения 0, 2, 4, 6

3) FOR X = 2,5 TO1 STEP — .5 — цикл выполняется 4 раза. X изменяется от 2.5 до 1 с шагом — 0,5, т. е. X принимает значения 2.5, 2, 1.5,1

Структура оператора NEXT:

NEXT <простая числовая переменная>

Параметр <простая числовая переменная> — это переменная цикла, указанная в операторе FOR.

Оператор NEXT задает конец цикла. Он сравнивает текущее значение переменной с предельным значением, т. е. со значением <а.в.2>, и если они отличаются меньше, чем на шаг (текущее значение стало конечным), то цикл кончается. Такое окончание цикла является естественным выходом из цикла. После NEXT будет выполняться следующий оператор программы.

Если текущее значение отличается от значения <а.в. 2> больше, чем на шаг, то оператор выполняет изменение значения переменной

цикла на шаг и последующий переход к заголовку цикла, т. е. следующим будет выполняться оператор, который записан в программе после FOR. Независимо от параметров заголовка цикл выполняется хотя бы один раз.

Выполнение цикла разрешается только от заголовка цикла, т. е. нельзя входить в цикл (выполнять переходы к операторам тела цикла), минуя заголовок. Это приводит к ошибке при выполнении оператора NEXT. В непосредственном счете весь цикл, включая операторы FOR, NEXT, должен входить в одну строку.

Пример 67.

```
10 FOR X=-.5TO0STEP.05
20 PRINT "X=",X,"Y=";SIN(X)
30 NEXT X
```

Пример 68.

```
10 INPUT "ВВЕДИТЕ КОЛИЧЕСТВО ЗНАЧЕНИЙ",K
20 PRINT "ВВЕДИТЕ ЗНАЧЕНИЯ ЧЕРЕЗ CR/LF"
30 FOR M=1TOK
40 INPUT X:PRINT /0C,X
50 NEXT M
```

Пример 69.

```
10 INPUT "ВВЕДИТЕ НАЧАЛЬНОЕ,КОНЕЧНОЕ ЗНАЧЕНИЕ И ШАГ ИЗМЕНЕ
НИЯ АРГУМЕНТА",A1,A2,A3
20 FOR X=A1TOA2STEPA3
30 M=M+1:REM СЧЕТЧИК ЗНАЧЕНИЙ В СТРОКЕ ПЕЧАТИ
40 PRINTUSING "#.### ",COS(X);
50 IF M<10THEN60:M=0:PRINT
60 NEXT X
```

Допускается досрочный выход из цикла, т. е. при числе повторов, меньшем заданного в заголовке цикла. Досрочный выход осуществляется при выполнении какого-либо условия оператором программного перехода.

Пример 70. Ввести К значений, но не больше 10. Если К не равно 10, то закончить набор значением «***».

```
5 DIM A$25
10 FOR K=1TO10
20 INPUT "ВВЕДИТЕ Ф.И.О. ИЛИ ***",A$
30 IF A$="***"THEN50:REM ДОСРОЧНЫЙ ВЫХОД ИЗ ЦИКЛА
40 PRINT A$:NEXT K
50 END
```

Два и более цикла, записанные в программе, могут занимать различное взаимное положение аналогично скобкам в арифметическом выражении.

Независимые циклы следуют один за другим. Вложенные циклы входят в тело другого цикла. Глубина вложения (степень вложения) для операторов цикла в языке БЕЙСИК ограничивается лишь наличием оперативной памяти в служебной зоне и числом переменных, которые можно использовать в качестве переменных цикла.

Недопустимо пересечение циклов.

Допустимый способ вложения

```
FOR A
  FOR B
    FOR C
    NEXT C
  NEXT B
  FOR K
  NEXT K
NEXT A
```

Недопустимый способ вложения

```
FOR A
  FOR B
    FOR C
  NEXT B
  NEXT C
NEXT A
```

Пример 71.

Распечатать 20 введенных значений в 4 строки по 5 значений.

```
10 FOR K=1TO4:REM ЦИКЛ ДЛЯ СТРОК
20 FOR M=1TO5:REM ЦИКЛ ДЛЯ СТОЛБЦОВ
30 INPUT X:PRINT /0C,X,
40 NEXT M:PRINT %0C:REM ПЕРЕВОД СТРОКИ
50 NEXT K
```

Пример 72.

Вычислить значения $Y = AX + B$, если для каждого A, принимающего значения 5, 12, 17; B принимает значения — 5, 0, 5; в свою очередь, для каждого B переменная принимает значения от —17до 18 с шагом 5.

```
10 DATA 5,12,17,-5,0,5
20 FOR K=1TO3
30 RESTORE K:READ A:PRINT "A=";A
40 FOR M=1TO3
50 RESTORE 3+M:READ B:PRINT "B=";B
60 FOR X=-17TO18STEP5
70 Y=A*X+B
80 PRINTUSING "-####",A*X+B,
90 NEXT X:PRINT
100 NEXT M:NEXT K
```

Использование массивов в циклах. На практике циклические процессы почти всегда связаны с использованием оперативной памяти

для записи значений в оперативную память или выборки значений из памяти для обработки. В связи с этим возникает вопрос переадресации переменных, т. е. организации автоматического использования значений различных переменных при одном обозначении их в программе. Такое обозначение возможно, если переменные — элементы одного массива.

Так, совокупность элементов A (1), A (2), A (3), ..., A (17) одномерного действительного массива A можно обозначить в виде A (K) и указать пределы допустимых значений K от 1 до 17 включительно, т. е. запись A (K) обозначает использование текущего K-го элемента массива A.

Для двумерного массива можно обозначить:

A (M, N) — текущий элемент в массиве;

A (5, N) — текущий элемент в пятой строке;

A (M,7) — текущий элемент в седьмом столбце.

Итак, в цикле для обозначения текущего элемента массива используется косвенная адресация, т. е. задание местоположения текущего элемента в массиве арифметическими выражениями.

Например, ввести и записать в оперативную память (запомнить) 20 чисел, а затем распечатать их построчно по пять чисел в строке.

Рассмотрим два варианта решения: с использованием одномерного и двумерного массивов.

Пример 73.

```
10 DIM A(20)
20 FOR K=1TO20
30 INPUT A(K):NEXT K
40 FOR K=1TO20
50 M=M+1
60 PRINT A(K);
70 IF M<5THEN80:PRINT :M=0
80 NEXT K
```

Пример 74.

```
10 DIM A(1,20)
20 FOR K=1TO20
30 INPUT A(1,K):NEXT K
40 MAT REDIM A(4,5)
50 PRINT A()
60 REM РАСПЕЧАТКА ЧАСТИ МАССИВА
70 MAT REDIM A(5,4)
80 FOR K=2TO4
90 FOR M=1TO3
100 PRINT A(K,M),
110 NEXT M:PRINT
120 NEXT K
```

Если в цикле указан один массив или несколько массивов с одинаковым способом изменения номера, то для задания номера элемента используется чаще всего переменная цикла. Если имеется несколько массивов и номера элементов в них изменяются неодинаково, то для одного массива используется переменная цикла в качестве номера элемента, а для другого специальная переменная — счетчик, значение которой изменяется программно.

Пример 75. Элементам массива B с четными номерами присвоить значения элементов массива A.

```
10 DIM A%(5),B%(1,10)
20 MAT READ A%
30 FOR K=1TO5
40 M=M+2:B%(1,M)=A%(K)
50 NEXT K:PRINT B%()
60 DATA 1,2,3,4,5

 0  1  0  2  0  3  0  4  0  5
```

Пример 76.

```
10 DATA 1,2,3,4,5
20 DIM A(5),B(10)
30 MAT READ A
40 FOR K=2TO10STEP2
50 B(K)=A(K/2)
60 NEXT K
70 PRINT B()
```

Пример 77.

Найти в двумерном массиве порядковый номер элемента, значение которого равно значению переменной. Решение возможно разными способами в зависимости от установленной размерности:

```
10 DIM A(2,5)
20 DATA 0,1,2,3,4,5,6,7,8,9
30 MAT READ A
40 INPUT "ВВЕДИТЕ ЧИСЛО ДЛЯ СРАВНЕНИЯ",B
50 FOR K=1TO2
60 FOR M=1TO5
70 IF A(K,M)=BTHEN100
80 NEXT M:NEXT K
90 PRINT "ЧИСЛА=";B,"В МАССИВЕ НЕТ":END
100 PRINT "ПОРЯДКОВЫЙ НОМЕР ЧИСЛА";B;"  В МАССИВЕ A РАВЕН ";
(K-1)*5+M
```

Пример 78.

```
10 DIM A(2,5)
20 DATA 0,1,2,3,4,5,6,7,8,9
30 MAT READ A
40 INPUT "ВВЕДИТЕ ЧИСЛО ДЛЯ СРАВНЕНИЯ",B
50 MAT READ A(10,1)
60 FOR K=1TO10
70 IF A(K,1)=BTHEN90:NEXT K
80 PRINT "ЧИСЛА ";B,"В МАССИВЕ НЕТ":END
90 PRINT "ПОРЯДКОВЫЙ НОМЕР ЧИСЛА ";B;"-";K
```

Примечание. Смену шага и значения, задающего конечное значение переменной цикла в теле цикла, изменить нельзя, т. е. обязательно после смены необходимо входить в цикл через заголовок.

4. ОРГАНИЗАЦИЯ ПОДПРОГРАММ И ОБРАЩЕНИЯ К НИМ

Подпрограмму можно определить как совокупность операторов, записанных подряд и оформленных в соответствии со стандартными требованиями к оформлению подпрограммы. Выполняются операторы подпрограммы многократно, но в отличие от цикла выполнение происходит лишь по «вызову», т. е. при обращении к подпрограмме. Обращение к подпрограмме — переход к началу подпрограммы, последующее исполнение операторов подпрограммы и возврат в основную программу для продолжения счета с оператора, следующего за оператором обращения. Подпрограмма может располагаться в любом месте программы (обычно все подпрограммы записываются в конце программы, а исполняется подпрограмма в том месте, где есть оператор обращения к ней).

БЕЙСИК допускает организацию подпрограмм двух типов:

1) *непомеченная подпрограмма* может содержать любые операторы языка, но последним обязательно должен быть оператор RETURN, выполняющий возврат из подпрограммы в основную программу; обращение к подпрограмме выполняется из программы лишь оператором GOSUB;

2) *помеченная подпрограмма* (имеющая метку) отличается от непомеченной по структуре наличием оператора DEFFN′ с номером подпрограммы от 0 до 255 (см с. 69).

Оператор DEFFN′ фиксирует начало подпрограммы, присваивает ей номер, указанный в операторе. Функционально помеченная подпрограмма отличается от непомеченной возможностью передавать в подпрограмму при обращении параметры подпрограммы, т. е. данные, используемые в подпрограмме для счета.

Для результатов, получаемых в подпрограмме, задание параметров при обращении к подпрограмме не предусмотрено.

Обращение к помеченной подпрограмме выполняется программно оператором GOSUB′ и с клавиатуры (вручную) клавишей СФ с номером, равным номеру подпрограммы. Обращение с клавиатуры возможно

лишь для подпрограмм с номерами 0—31, которые не имеют параметров, после того как был выполнен RUN CR/LF (запуск на счет программы). После выполнения подпрограммы машина переходит в режим ожидания, прерванный обращением.

Язык допускает использование вложенных подпрограмм, т. е. допускает обращение из одной подпрограммы в другую, которая и будет вложенной (исполняемой внутри первой подпрограммы). Выполнение внутренней подпрограммы завершается возвратом (оператором RETURN) во внешнюю подпрограмму, после чего продолжается выполнение внешней подпрограммы, которое завершается возвратом в основную программу. Ограничения на степень вложенности подпрограмм те же, что и для циклов — объем памяти, т. е. ограничений нет.

Взаимное расположение подпрограмм и циклов многообразно. Во-первых, из цикла допускается обращение к подпрограмме. В этом случае возврат из подпрограммы может быть как оператором RETURN (что логичнее и предпочтительнее), так и оператором NEXT, соответствующим заголовку цикла, из которого было обращение. Во-вторых, в подпрограмму может входить цикл. Этот цикл должен быть закончен в подпрограмме. Если в подпрограмме будет только заголовок цикла, то цикл будет выполняться от заголовка до оператора RETURN, так как RETURN закроет цикл (сотрет из служебной памяти машины сведения о нем как о текущем цикле). Поэтому пересечения циклов с подпрограммами нежелательны, хотя и не приводят к сбою.

Оператор GOSUB. Оператор обращения к непомеченной подпрограмме имеет параметр — номер строки:

GOSUB <номер строки>

Оператор GOSUB предназначен для обращения к подпрограмме. При выполнении оператор GOSUB запоминает в служебной зоне памяти адрес следующего за ним оператора, который далее будет использован оператором RETURN для возврата из подпрограммы, затем оператор GOSUB выполняет переход к подпрограмме по номеру строки. Эта строка — первая строка подпрограмм. Она не должна быть внутри цикла. Оператор GOSUB не может быть последним в программе и не используется в непосредственном счете.

Пример 79.

```
. . .
70 A = 17.5 : B = 250
80GO SUB 140 : PRINT X
. . .
140 REM НАЧАЛО ПОДПРОГРАММЫ
200 PETURN : REM ВОЗВРАТ НА ОПЕРАТОР PRINT X
```

Оператор GOSUB'. Структура оператора:

COSUB' < номер подпрограммы> [(<список параметров>)]
<список параметров> : := <фактический параметр> [{,<фактический параметр>}...]
<фактический параметр> : := <а.в.> | <символьная переменная>| <символьная константа>

Оператор GOSUB′ предназначен для обращения к помеченной подпрограмме по номеру подпрограммы.

При выполнении оператор GOSUB′ запоминает адрес возврата из подпрограммы для RETURN, отыскивает в программе DEFFN′ с номером подпрограммы, указанным в операторе обращения, и присваивает формальным параметрам списка оператора DEFFN′ значения фактических параметров списка оператора GOSUB′. Поэтому требуется четкое соответствие фактических параметров формальным параметрам по типу и количеству. После этого выполняются операторы подпрограммы и осуществляется возврат в программу оператором RETURN.

Если в операторе DEFFN′ не указан список параметров, то не должно быть списка параметров и в операторе GOSUB′.

Оператор GOSUB′ не должен быть последним в программе.

Пример 80.

```
. . .
 70 GOSUB' 5 (X, Y %, A¤) : PRINT K
. . .
150 GOSUB' 5 (C, K %, B ¤ )
 160 IF C2 <THEN 100
. . .
 550 DEFFN' 5 (A, B, C¤)
. . .
600 RETURN
```

Возврат выполняется после первого обращения к оператору RRINT в 70 строке, а при втором обращении к оператору IF — в 160 строке.

Оператор RETURN. Оператор не имеет параметров.

RETURN — оператор возврата из подпрограммы состоит только из имени. Для выполнения перехода (возврата) к месту обращения оператор использует адрес оператора возврата из служебной зоны памяти, куда этот адрес заносится операторами **GOSUB** и **GOSUB′**, при этом всегда используется адрес, занесенный последним. Использованный адрес оператор RETURN уничтожает (стирает в служебной зоне). Досрочные выходы из подпрограммы, а также использование GOSUB вместо GOTO накапливают адреса возврата, что может привести к переполнению памяти (магазина адресов). Оператор RETURN стирает также информацию о всех циклах, открытых и незаконченных в подпрограмме (закрывает их вместо NEXT). В непосредственном счете не используется.

Пример 81.

Вычислить значения выражения

$$f(x, y)=\sqrt{ax^2+by^2+K}$$

$$K=\begin{cases} a\sin(x)+b, & \text{если } |x|>|y|; \\ b\sin(y)+a, & \text{если } |x|<|y|; \\ (a+b)/2, & \text{если } |x|=|y|. \end{cases}$$

$a = 17{,}5$; $b = 0{;}75$; значения x и y переменные, вводимые попарно с клавиатуры.

```
10 A=17.5:B=0.75
20 INPUT "ВВЕДИТЕ X,Y",X,Y
30 P=ABS(X)-ABS(Y)
40 ON SGN(P)+2GOSUB70,90,100
50 F=SQR(A*X↑2+B*Y↑2+K)
60 PRINT "F(X,Y)= ";F:GOTO 20
70 GOSUB   1(B,Y,A):REM ABS(X)<ABS(Y)
80 RETURN
90 K=(A+B)/2:RETURN :REM ABS(X)=ABS(Y)
100 GOSUB   1(A,X,B):REM ABS(X)>ABS(Y):RETURN
120 DEFFN   1(E,T,H):REM ВЛОЖЕННАЯ ПОДПРОГРАММА ВЫЧИСЛЕНИЯ K
130 K=E*SIN(T)+H
140 RETURN
```

Оператор RETURN CLEAR. Оператор не имеет параметров.

RETURN CLEAR — оператор стирания адреса возврата из подпрограммы в служебной зоне оперативной памяти. Используется после досрочного выхода из подпрограммы (не через RETURN) для предупреждения переполнения зоны памяти адресов возврата.

Пример 82.

```
70 GOSUB' 1
...
200 DEEFN' 1
210 IF K( )2 * X — Y THEN 300 — досрочный выход из подпрограммы;
...
290 RETURN                  — возврат в программу;
300 RETURN CLEAR            — стирание адреса возврата при досрочном выходе из цикла
...
```

5. ПОЛУЧЕНИЕ СПРАВОЧНОЙ ИНФОРМАЦИИ О ПРОГРАММЕ ОПЕРАТОРОМ LIST

Оператор LIST имеет четыре формы, позволяющие:

вывести текст программы полностью, страницами, выборочно по заданному диапазону строк и по одной строке;

получить справочные данные о помеченных подпрограммах и обращениях к ним;

получить справочные данные о программных переходах в программе (обо всех переходах или о переходах, задаваемых параметрами оператора LIST);

получить справочные данные об использовании переменных в программе; справку можно получить обо всех переменных или о переменных, заданных параметрами оператора LIST.

Вывод справочной информации оператор LIST выполняет на устройство, заданное в операторе или (по умолчанию) на БОСГИ, или на устройство, заданное оператором SELECT LIST.

Вывод текста программы. Оператор имеет следующую структуру:

```
LIST [S] { [⟨устройство⟩]
         { [⟨устройство⟩, ] { ⟨номер строки 1⟩ [, ⟨номер строки 2⟩]
                            { , ⟨номер строки 2⟩
```

Оператор LIST (форма 1) выводит текст программы на экран или печать полностью или частично.

Параметр <номер строки 1> задает номер выводимой строки, в сочетании с параметром <номер строки 2> — номер первой выводимой строки, а <номер строки 2> — номер последней выводимой строки.

Если параметр <номер строки 1> опущен, то <номер строки 2> — номер последней выводимой строки. Программа выводится сначала.

Параметр *S* указывает на вывод по страницам, при этом вызов каждой следующей страницы, начиная со второй, выполняется нажатием на клавишу CR/LF. Возможна следующая форма записи:

1) LIST — вывод всего текста программы;
2) LIST S— вывод первой страницы текста программы;
3) LIST 70 — вывод строки 70;
4) LIST, 70 — вывод строк программы с начала до строки 70 включительно;
5) LIST 20, 70 — вывод строк с 20 до 70 включительно.

Получение справочной информации о подпрограммах. Используется форма 2 оператора LIST:

LIST [<устройство>,]' [<номер подпрограммы>]

Оператор LIST (форма 2) выдает справку о помеченных подпрограммах, если параметр <номер подпрограммы> опущен, или о подпрограмме, номер которой указан в операторе. Справка о каждой подпрограмме выдается в виде:

```
{<номер строки с оператором DEFFN' >} — DEFFN' <номер подпрограммы>
{  ????                             }
[<список номеров строк с оператором GOSUB' > ]
```

Возможны следующие виды индицируемой информации при выполнении операций:

1) LIST/05,'
0210 — DEFFN'2
0070 0160 — к подпрограмме 2 есть обращения в строках 70 и 160;
2) LIST'1
???? — DEFFN'1
0019 — к подпрограмме 1, отсутствующей в программе, есть обращение в строке 19;
3) LIST'3
0220 — DEFFN'3 — к подпрограмме 3 нет обращений в программе.

Получение справочной информации о программных переходах. Используется форма 3 оператора LIST:

LIST [<устройство>,]¤ { <номер строки 1>, [<номер строки 2>] } / { [<номер строки 1>] [, номер строки 2>] }

Оператор LIST (форма 3) выдает справку об операторах, выполняющих программные переходы по номеру строки. Если все параметры после ¤ в операторе LIST отсутствуют, то выдается справка о всех переходах.

Если указан <номер строки 1> без последующих параметров, выдается справка о переходах к этому номеру; если указана еще запятая — справка о переходах к строкам в диапазоне от указанной строки до конца программы; если указаны все параметры — справка о переходах в диапазоне указанных номеров строк; если указаны запятая и параметр <номер строки 2> — справка о переходах в диапазоне от начала программы до указанной строки.

Пример. Справки о переходе к строке 70:

```
LIST ¤ 70
   0070 — 0040 0160 0220
```

В строках 40, 160, 220 есть программные переходы к строке 70.

Получение справочной информации об используемых переменных. Используется форма 4 оператора LIST:

LIST [⟨устройство⟩,] V [⟨список LIST⟩]

⟨**список LIST**⟩ : : = { ⟨параметр LIST 1⟩, [⟨параметр LIST2⟩] } / { [⟨параметр LIST 1⟩] [, параметр LIST2⟩] }

⟨**параметр** LIST { 1 / 2 }⟩ : : = { ⟨простая числовая переменная⟩ / ⟨простая символьная переменная⟩ / ⟨метка массива⟩ }

Оператор LIST (форма 4) выдает справку об используемых в программе переменных (в каких строках они записаны).

Оператор LIST без параметров после V выдает справку о всех переменных в последовательности о целых (по алфавиту), о действительных, о символьных. Эту последовательность следует учитывать, задавая в операторе диапазон интересующих переменных параметрами LIST. Задание диапазона аналогично заданию диапазона в операторе LIST ¤.

Пример 83.

```
   LIST V
A % — 40 0070 0090 0200
X% — 0050 0070
   A — 0010 0060
```

6. ОТЛАДКА ПРОГРАММЫ

Процесс поиска и устранения ошибок в программе называется отладкой программы. Для отладки используются:

просчет контрольного примера;

печать и анализ промежуточных результатов при счете по программе;

оператор STOP и клавиша CONTINUE;

пооператорный режим;

режим трассирования.

Пооператорный режим счета. Пооператорный режим задается с помощью клавиши HALT/STEP. Нажатие на клавишу HALT/STEP останавливает выполнение программы после завершения исполняемого в данный момент оператора.

Запуск программы на счет при отладке осуществляется оператором GOTO <номер строки> CR/LF, где <номер строки> — начало участка отладки. При нажатии на клавишу HALT/STEP на экране индицируется строка программы полностью и двоеточие с курсором (:—). Следующее нажатие на клавишу HALT/STEP стирает с экрана выполнившийся оператор, оставшаяся часть строки сохраняется, и так до тех пор, пока не исчезнет вся строка программы.

Если в строке были ошибочные операторы, то индицируется сообщение об ошибке ERR с указанием ее номера. Продолжить нормальное исполнение программы можно нажатием на клавишу CONTINUE.

Если участок отладки находится в середине программы, то перед ним ставится оператор STOP, и программа запускается на счет с начала оператором RUN CR/LF. При выполнении оператор STOP вызывает останов программы, после чего задается пошаговый режим.

С помощью дополнительных операторов PRINT, включенных в программу, можно получить промежуточные значения переменных, т. е. те значения, которые принимают переменные в процессе счета по программе.

Режим трассирования. Промежуточные значения переменных, а также встречающиеся в программе переходы по номерам строк можно просмотреть в режиме трассирования (слежения). При этом каждый раз автоматически выводится имя переменной, принятое ею значение и сообщение о программном переходе в виде

```
30 GOTO 100
TRANSFER TO 100
```

Начало участка трассирования задается оператором TRACE, конец — TRACE OFF.

Режим трассирования может быть использован самостоятельно. Тогда участок трассирования ограничивается включенными в программу операторами TRACE и TRACE OFF. Если результаты трассирования просматриваются на экране, то задается пауза индикации оператором SELECT P [(<цифра>)].

Если нужно получить протокол трассирования, то в качестве консольного устройства вывода назначается ПУ оператором SELECT CO 0C [(<длина строки>)].

Режим трассирования может быть использован также с пооператорным режимом. В этом случае сначала ограничивается участок трассирования, а затем в нем задается пошаговый режим.

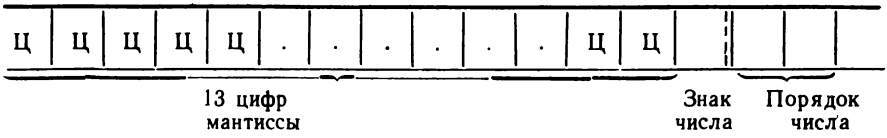
Глава X
ФОРМАТЫ ПРЕДСТАВЛЕНИЯ ИНФОРМАЦИИ

Символы, предназначенные для представления символьной и цифровой информации в зоне значений машины, для носителей информации и устройств ввода-вывода кодируются в соответствии с таблицей КОИ-8 (ГОСТ 19768-74).

1. ПРЕДСТАВЛЕНИЕ ДАННЫХ В ЗОНЕ ДАННЫХ

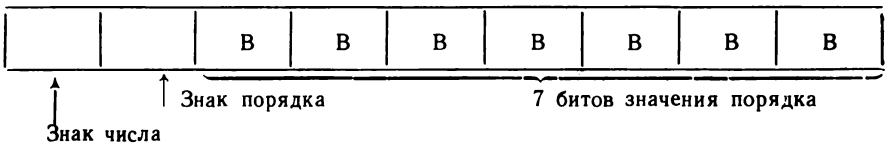
В зоне данных действительные, целые и символьные значения представляются следующим образом.

Действительное значение



Здесь Ц — цифра мантиссы (4 бита в двоично-десятичном коде) Число цифр формата соответствует разрядности (13).

Знак и порядок числа



Здесь В — двоичная цифра 0 или 1 (1 бит).

Знак числа может принимать одно из двух значений:

0, что означает знак плюс (+),

1, что означает знак минус (—).

Знак порядка также может принимать одно из двух значений:

0, что означает знак плюс (+),

1, что означает знак минус (—).

При положительном порядке значение порядка представлено в прямом коде, при отрицательном — в дополнительном.

Значение порядка занимает 7 бит в двоичном коде.

Формат числа + 1.234567800000E + 91 имеет вид:

0001	0010	0011	0100	0101	0110	0111	1000	0000	...	0000	0101	1011

Целое значение (—7999 — + 7999)

1-й разряд								16-й разряд	
Знак числа	В	В		.		.		.	В	В

Здесь В — двоичная цифра 0 или 1 (1 бит). Всего 16 двоичных разрядов. Знак числа может принимать одно из двух значений (0 или 1).

Целое число + 6167 имеет вид:

0	110	0001	0110	0111

Символьное значение

С	С	.	.	.	С	С

Здесь С — символ в коде КОИ-8 (1 байт).

Пример. Формат слова ПРИМЕР имеет вид:

Код символа	11110000	11110010	11101001	11101101	11100101	11110010
Символ	П	Р	И	М	Е	Р

2. ПРЕДСТАВЛЕНИЕ ФАЙЛОВ ДАННЫХ И ПРОГРАММ НА НОСИТЕЛЕ

Структура файла

Открывающий блок файла	1-й блок файла		*n*-й блок файла	Закрывающий блок файла

Длина блока (физической записи) устанавливается на входном языке машины (длина строки).

Открывающий блок файла

НЗ	С	С	...	С	ПЗ	П	...	П	КС

Здесь НЗ — идентификатор блока в коде КОИ-8;

С — символ (код КОИ-8), если имя файла отсутствует, то первые восемь байтов заполняются кодами «пробел»;

ПЗ — признак защиты в коде КОИ-8;

П-символ «пусто» в коде КОИ-8;
КС — контрольная сумма.

Информационный блок файла

И	РЭ	Э	РЭ	Э	РЭ	Э	РЭ	П	...	П	П	КС

Здесь РЭ — длина элемента и его тип (символьный, цифровой или программная строка), занимает 2 байт;
П — код «пусто» в коде КОИ-8;
КС — контрольная сумма, один байт;
И — идентификатор в коде КОИ-8;
Э — элемент, т. е. одно данное; число занимаемых байтов соответствует формату данного.
Типы идентификаторов:
НТ — начало логической записи;
КТ — конец логической записи;
СП1 — очередной блок;
СП2 — блок, содержащий целиком логическую запись;
НЗ — начало открывающего блока файла.

Закрывающий блок файла

РФ	П	П	...	П	КС	на МЛ

РФ — разделитель файлов в коде КОИ-8;

РФ	К	П	П	...	П	КС	на диске

К — количество занятых файлом секторов.

Глава XI

ОБМЕН ИНФОРМАЦИЕЙ С НАКОПИТЕЛЕМ НА МАГНИТНЫХ ДИСКАХ

1. ОБЩИЕ СВЕДЕНИЯ

При программировании обмена информацией с НМД и НГМД следует учитывать ФАУ устройств и их технические характеристики: общее число секторов, число сторон диска для доступа (для НГМД одна или две) и нумерацию секторов. Только эти параметры определяют отличия обмена. Подготовка дисков к работе, программная организация обмена, использование дисковых операторов для НМД и НГМД принципиально одинаковы.

БЕЙСИК допускает организацию обмена с дисковыми устройствами в режиме адресации секторов и в режиме каталогизации. Режим каталогизации предусматривает определенную стандартную структуру формируемой информации на диске и соответствующий способ обработки.

Режим каталогизации отличается от режима адресации большей надежностью, т. е. большей степенью защиты информации от случайного искажения вследствие программных ошибок пользователя. В режиме адресации секторов распределение секторов выполняется пользователем при программировании (структура информации произвольна), а в режиме каталогизации распределение секторов автоматическое (системное), информация имеет файловую структуру по стандартному формату (см. с. 116).

2. ОБМЕН ИНФОРМАЦИЕЙ С ДИСКОВЫМИ УСТРОЙСТВАМИ В РЕЖИМЕ КАТАЛОГИЗАЦИИ

Структура информации на каталогизированном диске. Каталогизированный диск имеет две или три информационные зоны.

Первая зона — указатель каталога файлов (УКФ). УКФ размещается с начала диска (с нулевого сектора) и занимает 24 сектора (стандартная длина, устанавливаемая системой) или число секторов, заданное пользователем в операторе SCRATCH DISK.

УКФ содержит информацию о файлах, расположенных в зоне каталога, и общую информацию о состоянии диска: размер зон, первый свободный сектор в зоне каталога.

Вторая зона — каталог файлов (КФ) — располагается на диске за УКФ и занимает либо оставшуюся часть диска до конца, либо до сектора, указанного пользователем в операторе SCRATCH DISK.

КФ содержит информационные файлы: программные файлы и файлы данных.

Третья зона, наличие которой на каталогизированном диске необязательно, — рабочая, содержащая временные (рабочие) файлы данных, сведения о которых не заносятся в УКФ. Однако работать с файлами этой зоны можно так же, как с файлами зоны УКФ.

Итак, организация информации на диске напоминает сборник рассказов, а УКФ аналогичен оглавлению, расположенному в начале сборника. Просмотреть (распечатать) содержимое УКФ можно оператором LIST DC.

Форматизация и каталогизация диска. Новый диск или диск, информация на котором обновляется полностью, необходимо форматизовать и каталогизировать.

Форматизация диска, т. е. разбиение его на секторы с записью служебной информации, выполняется оператором PRINT. Дисковое устройство, на котором выполняется форматизация диска, задается либо в самом операторе PRINT физическим адресом: PRINT/18 ..., PRINT/1C ..., либо предварительным выполнением оператора SELECT PRINT 18 (256) или ELECT PRINT 1C (256) соответственно для НГМД и НМД.

Последовательность действий при форматизации гибкого диска:
установить диск в НГМД F (НГМД R);
снять клавишей защиту записи устройства F (R);
выполнить в непосредственном счете оператор

```
PRINT/18, HEX (1B000000000400);
или
PRINT/18, HEX (1B80000000049F);
```
} для устройства F.

```
PRINT/18, HEX (1B000100000400);
или
PRINT/18, HEX (1B80010000049E);
```
} для устройства *R*.

Символ ";" в конце оператора является обязательным (набираемым) разделителем.

Для НМД форматизация выполняется аналогично, изменяется ФАУ 18 в операторе на ФАУ 1С.

При выполнении оператора PRINT каждый сектор заполняется символами «пусто» (HEX (00)) и заносится байт контрольной суммы информации в секторе.

Контроль (верификация) качества диска выполняется оператором VERIFY, имеющим следующую структуру:

VERIFY <тип диска> [<устройство>], (<а.в. 1>), <а.в.2>)

<тип диска> : := F|R|T, T — любой, т. е. тот, который задан оператором SELECT DISK.

Параметр <а.в.1> задает начальный сектор, а параметр <а.в.2> — конечный сектор контролируемой зоны секторов на диске.

Контроль выполняется считыванием содержимого каждого сектора и сравнением контрольных сумм сектора считанной и вычисленной.

Если секторы не указаны, то контролируются все секторы. Если параметр <устройство> не задан, то контролируется диск на НГМД или на устройстве, заданном оператором SELECT DISK.

При обнаружении ошибок на контрольное устройство вывода выдается информация о номере ошибочного сектора:

ERROR IN SECTOR <номер сектора>

Оператор VERIFY используется на любом этапе обработки.

Запись оператора имеет вид:

VERIFY F/1C, (10, 1500) — секторы 10 — 1500 на 1CF,
VERIFY R (200, 700) — секторы 200—700 на 18R.
VERIFY F — секторы 0 — 1000 на 18F.

Каталогизация диска (деление диска на зоны).

Для подготовки диска к работе в режиме каталога необходимо форматизованный диск (контроль не обязателен) установить на соответствующее устройство, снять защиту записи и выполнить в непосредственном счете оператор SCRATCH DISK:

SCRATCH DISK <тип диска> [<устройство>,] [LS = <а.в.1>,] END = <а.в.2 >

Если в операторе устройство не указано, то при выполнении оператора обращение происходит к устройству, заданному оператором SELECT DISK, выполненным ранее. Если SELECT DISK не выполнялся, то обращение происходит к НГМД (ФАУ 18), т. е. по умолчанию обращение происходит к НГМД.

Параметр LS и целая часть значения <а.в.1> задает размер зоны УКФ. Если параметр опущен, то для УКФ резервируются 24 сектора с 0-го по 23-й. Максимально допустимое число секторов для УКФ составляет 255.

Число резервируемых секторов определяется пользователем в соответствии с числом предполагаемых для записи на диск файлов: в каждый сектор зоны УКФ записывается информация о 16 файлах, а в нулевой сектор — о 15 файлах.

Секторы зоны УКФ при выполнении оператора SCRATCH DISK заполняются кодами пусто (HEX (00)).

Информация о файле в зоне УКФ имеет следующую структуру:

2 байт — тип файла (P-программный, D — файл данных, SP и SD— ликвидируемые файлы);

2 байт — начальный адрес (номер) сектора зоны КФ, с которого зарезервировано место для файла;

2 байт — конечный адрес сектора резерва файла;

8 байт — имя файла.

Параметр END задает границу зоны КФ диска.
Значение параметра <а. в.2> — адрес последнего сектора зоны КФ. Он не должен задаваться бóльшим последнего адреса сектора на диске.

Запись оператора имеет вид:

1) SCRATCH DISK F END = 1000 — на гибком диске, установленном на 18F, зарезервированы под УКФ секторы с 0-го по 23-й и под КФ секторы с 24-го по 1000-й, т. е. до конца диска;

2) SCRATCH DISK R/1C, LS = 30, END = 7000 — на съемном жестком диске (1CR) зарезервированы секторы с 0-го по 29-й для УКФ и с 30-го по 7000-й для КФ;

3) SELECT DISK 1C R:

SCRATCH DISK T LS = 17, END = 4000
SCRATCH DISK FLS = 25, END = 8000

Здесь выполнена каталогизация съемного и фиксированного диска НМД с ФАУ 1С, при этом секторы распределились по зонам диска следующим образом:
на 1C R секторы 0—16—УКФ, 17—4000 — КФ;
на IC F секторы 0—24 — УКФ, 25 — 8000 — КФ.

Изменение зоны каталога файлов оператором MOVE END. Зону каталога файлов можно увеличить или уменьшить в процессе работы с дисковыми устройствами, т. е. задать граничным сектор с адресом, бóльшим или меньшим адреса прежнего граничного сектора зоны КФ. Уменьшать зону следует с осторожностью, так как возможна потеря информации зоны КФ.

Изменение зоны КФ выполняется оператором MOVE END:

MOVE END ⟨тип диска⟩ [⟨устройство⟩,]=⟨а. в.⟩

Целая часть арифметического выражения задает новый адрес последнего сектора зоны КФ. Зону УКФ изменять нельзя.

Получение справочной информации о каталогизированном диске. Так как зона УКФ содержит информацию о всех файлах, записанных в зоне каталога, то чтение и просмотр содержимого УКФ позволяют как

бы увидеть, что за информация записана на диске: сколько файлов, какого типа файлы, резерв секторов каждого файла и фактически использованное число секторов, место расположения файла в зоне каталога, граница зоны каталога и адрес первого свободного сектора в зоне КФ. Оператор LIST DC выдает всю эту информацию на экран или ПУ.

Структура оператора LIST DC имеет вид:

LIST DC тип диска [[⟨устройство⟩] | [⟨устройство⟩,] { символьная переменная | символьная константа }]

Оператор LIST DC предназначен для получения справочной информации о каталогизированном диске.

Оператор LIST DC использует два устройства: дисковое устройство—устройство ввода (откуда получается справка); БОСГИ или ПУ — устройство вывода (куда выдается справка с диска).

В конструкции оператора параметром <устройство> и параметром <тип диска> задается дисковое устройство.

Если параметр <устройство> не задан, берется справка с гибкого диска — устройства с ФАУ 18 или с устройства, заданного оператором SELECT DISK.

Устройством вывода по умолчанию является БОСГИ, т. е. справка индицируется на экране. Для печати справки устройство вывода (печатающее устройство с ФАУ 0С) следует задать оператором SELECT LIST.

Наличие в операторе символьной переменной или символьной константы позволяет просматривать каталог с заданным условием, так как значение переменной или константы является маской для имени файла в справке. Например, если указана константа «_ _К», то на экране будут индицироваться имена файлов, у которых третьим символом является буква К. Символ пробел в маске означает, что значение символов в данной позиции имени файла может быть любым. Если длина символьной переменной больше 8 байт, то маской служат ее первые 8 байт.

Если константа или переменная в операторе не указаны, то выдается полная справка о всех файлах.

Запись оператора имеет вид:

```
1) LIST DC F
2) LIST DC R A ¤
3) LIST DC R/1C, «ИСХ»
4) SELECT LIST 0C:SELECT DISK 1C F:LIST DC F
5) SELECT LIST 0C:LIST DC F/1C
```

В примерах 1, 2, 3 справка выдается на БОСГИ, в 4 и 5 — на печать с жесткого диска.

Справка содержит следующую информацию:

NAME — имя файла;

TYPE — тип файла (P-программный, D — данных);

START — начальный адрес сектора файла;

END — конечный адрес сектора файла;

USED — число секторов, использованных в файле;
INDEX SECTORS — число секторов в УКФ;
END CATALOG AREA — последний сектор в КФ;
CURRENT END — первый свободный сектор в КФ.

Организация обмена информацией с дисковыми устройствами. Для каталогизированного диска используется файловый формат информации (см. с. 116). Файлы размещаются в зоне каталога файлов. Организуются файлы двух типов: программные и файлы данных.

Программные файлы оформляются стандартно по формату системой БЕЙСИК при выполнении оператора вывода программы на диск SAVE DC. Считывается и загружается программа с диска полностью при выполнении оператора LOAD DC (считывается весь программный файл). Для записи и считывания программного файла используется только имя файла.

Организация *файла данных* на диске и чтение информации из него выполняются пользователем программно. Совокупность дисковых операторов языка позволяет формировать файл по частям, т. е. выводить информацию порциями — логическими записями. Разрешается дозапись в файл, т. е. вывод новых логических записей в файл, который уже есть на диске. Чтение информации из файла возможно и полностью, и по частям (порциями). В общем случае порция чтения не равна логической записи (порции вывода).

Ввод и вывод данных в файл возможен, когда файл—рабочий (открытый). Одновременно открытыми могут быть до восьми файлов. Все открытые файлы нумеруются от 0 до 7. Использование номеров произвольное. Открывается файл по имени и получает при этом номер, а затем обращение к этому файлу (при вводе, выводе, перемещениях и т. д.) выполняется по номеру файла. Открытые файлы могут размещаться как на одном диске, так и на нескольких дисках. Под одним номером может быть открыт только один файл. Номер файла для дисковых устройств имеет еще одно функциональное назначение — он задает дисковое устройство, на котором находится открытый файл, т. е. номер файла одновременно является и логическим номером дискового устройства. Записывается номер файла в дисковых операторах так же, как логический номер устройства после символа #.

Например, DATA SAVE DC F # 5, A( ) — эта запись означает, что в файл с номером 5, находящийся на устройстве, логический номер которого 5, выводится действительный массив A.

При программировании следует учитывать требования, предъявляемые к использованию номера. Сначала номер рассматривается как логический номер устройства, т. е. оператором SELECT # ..., ему ставится в соответствие ФАУ устройства, а затем этот номер рассматривается как номер файла, т. е. ему ставится в соответствие имя файла, открываемого операторами DATA SAVE DC OPEN и DATA LOAD DC OPEN.

Для реализации работы с несколькими открытыми файлами в одной программе в системе БЕЙСИК используется таблица устройств.

Эта таблица формируется в служебной зоне памяти и используется системой автоматически. Программисту-пользователю важно знать, как она используется, чтобы не испортить и правильно использовать информацию на диске. В таблице имеются восемь строк с номерами от 0 до 7. Каждая строка таблицы содержит информацию о текущем состоянии открытого файла с номером, равным номеру строки таблицы.

В информацию о файле входит: физический адрес дискового устройства (ФАУ) и тип диска, определяющие размещение открытого файла; адрес начального сектора файла; адрес последнего сектора файла; адрес текущего сектора файла.

При работе с одним файлом номер файла в дисковых операторах можно не указывать, система считает его нулевым. Сведения о файле записываются в нулевую строку таблицы.

Если файл размещается на гибком диске, то можно не указывать и ФАУ устройства, т. е. не выполнять оператор SELECT #, так как в нулевую строку таблицы автоматически записывается ФАУ 18 и тип диска F при начальной генерации систем БЕЙСИК и при выполнении операратора CLEAR. Это и означает, что для дисковых операторов задан автоматически накопитель на гибких дисках типа F.

Итак, файл считается открытым, если сведения о нем записаны в таблицу устройств (т. е. файл принят системой на обслуживание в режиме каталога).

В организации обмена данными с дисковыми устройствами можно выделить три основных этапа:

1) открытие файла — запись сведений о файле в таблице устройств;

2) выполнение обмена — вывод данных в файл или ввод данных из файла в оперативную память, выполнение перемещений в файле и т. д.;

3) закрытие файла — стирание сведений о файле из таблицы устройств.

Конкретные ситуации, возникающие при обмене, требуют более детального подхода к организации обмена.

Для *открытия файла* необходимо:

выбрать номер для файла (0—7);

указать системе, на каком дисковом устройстве открывается файл с выбранным номером, записав в программе оператор SELECT # ... (SELECT # 018 F можно не указывать);

указать, с каким именем открывается файл оператором DATA SAVE DC OPEN, если файла нет на диске, или оператором DATA LOAD DC OPEN, если файл есть на диске.

Для *закрытия файла* можно использовать один из трех вариантов:

выполнить оператор DATA SAVE DC CLOSE <номер файла> или оператор DATA SAVE DC CLOSE ALL для закрытия всех открытых файлов, что соответствует стиранию одной строки таблицы устройств и стиранию информации всей таблицы;

открыть с этим номером другой файл, что соответствует замене сведений о файле в строке таблицы устройств;

выполнить оператор CLEAR, т. е. в программе файл можно не за-

крывать, он закроется при очистке памяти перед загрузкой другой программы.

Операторы открытия файла. К ним относятся операторы DATA SAVE DC OPEN, DATA LOAD DC OPEN.

Оператор DATA SAVE DC OPEN имеет две формы, предназначенных соответственно для открытия файла в зоне каталога и для открытия временного файла за зоной каталога.

Структура оператора (форма 1):

DATA SAVE DC OPEN <тип диска> [¤] [# <номер файла>,] ({<а.в.>} / {<имя 1>}) < имя>

Оператор DATA SAVE DC OPEN (форма 1) предназначен для открытия нового файла на диске в зоне каталога в свободной его части или на месте прежнего ликвидируемого файла. При этом происходит запись (SAVE) на диск сведений о новом файле в зону УКФ; которую можно прочитать оператором LIST DC.

Если в круглых скобках указан параметр <а. в.>, то в зоне каталога с адреса первого свободного сектора (CURRENT END) резервируется под файл место, т. е. число секторов, равное значению арифметического выражения без дробной части и знака. Изменяется и значение CURRENT END.

Если указан параметр <имя 1>, то для нового файла резервируется место на месте файла с именем, указанным в параметре <имя 1>, т. е. для нового файла резервируется столько секторов, сколько занимал прежний файл. Значение CURRENT END не изменяется. Имя прежнего файла может быть любым, т. е. совпадающим или несовпадающим с новым именем, лишь бы файл имел статус ликвидируемого (TYPE SD или TYPE SP, т. е. тип прежнего файла безразличен). Если файл не является ликвидируемым, будет ошибка (ERR 71).

Параметр <имя> — это имя нового файла, задаваемое символьной константой или символьной переменной, которое записывается в УКФ на диск.

Если параметр # <номер файла> отсутствует, то номер файла считается нулевым. Наличие символа ¤ в операторе означает, что вывод сведений о файле на диск будет выполняться с последующим контрольным считыванием для проверки правильности вывода по совпадению контрольных сумм.

После определения места на диске для нового файла в строку таблицы устройств, номер которой равен номеру файла, заносится из УКФ адрес (номер) начального сектора файла; адрес конечного сектора файла и адрес текущего сектора файла устанавливается равным адресу начального сектора файла.

Пример 84.

```
10 SELECT #018R,#118F,#218R
20 DATA SAVE DC OPEN R(22)"ИСХОД 1"
30 DATA SAVE DC OPEN F#1,(15)"ИСХОД 2"
40 DATA SAVE DC OPEN R#2,("ИТОГ")"ИТОГ 1"
```

Открыто три файла:
с номером 0 файл «ИСХОД 1» открыт на НГМД R;
с номером 1 файл «ИСХОД 2» открыт на НГМД F;
с номером 2 файл «ИТОГ 1» открыт на месте файла «ИТОГ» на НГМД R.

Структура оператора (форма 2):

DATE SAVE DC OPEN <тип диска> [¤] [# <номер файла>,]
TEMP, <адрес начального сектора>, <адрес конечного сектора>

Оператор DATA SAVE DC OPEN (форма 2) предназначен для открытия временного файла за зоной каталога файла. Резерв секторов для файла указывается параметрами <адрес начального сектора> и <адрес конечного сектора>.

Значения указанных адресов ограничиваются значением адреса последнего сектора в каталоге (в справке, полученной оператором LIST.DC, это параметр END CAT. AREA) и последним физическим адресом диска.

При выполнении оператора информация о временном файле без имени заносится в строку таблицы устройств с номером, равным номеру файла.

Пример 85.

```
10 SELECT #21C,#01CR
20 DATA SAVE DC OPEN R#2,TEMP,7200,7290
30 DATA SAVE DC OPEN TTEMP,7291,7359
```

На жестком съемном диске (1CR) открыты два временных файла с номерами 2 и 0.

Предполагается, что сектор 7200 расположен за зоной каталога.

Оператор DATA LOAD DC OPEN также имеет две формы и предназначен для открытия файла, имеющегося на диске.

Структура оператора (форма 1):

DATA LOAD DC OPEN <тип диска> [# номер файла>,] <имя>

Оператор DATA LOAD DC OPEN (форма 1) предназначен для открытия файла, сведения о котором есть в УКФ диска, т. е. это файл, в котором уже записаны данные, или файл, который только был ранее открыт оператором DATA SAVE DC OPEN.

Номер файла 0 можно в операторе не указывать. При выполнении оператора на диске (в УКФ) отыскивается файл с указанным именем, сведения о нем считываются из УКФ и записываются в строку таблицы устройств с номером, равным номеру файла.

Если имя файла не найдено, то будет ошибка (ERR 73), если файл ликвидируемый (TYPE SD), то будет ошибка (ERR.72).

Структура оператора (форма 2):

DATA LOAD DC OPEN <тип диска> [# <номер файла>,] TEMP,< адрес начального сектора>, <адрес конечного сектора>

Оператор DATA LOAD DC OPEN (форма 2) предназначен для открытия временного файла, записанного за зоной каталога. При выполнении оператора в таблицу устройства заносятся параметры файла.

Пример 86.

```
10 SELECT #11C
20 DATA LOAD DC OPEN R"ИТОГ 1"
30 DATA LOAD DC OPEN R#1,TEMP,7200,7290
```

Открыты те же файлы (только с другими номерами), которые были открыты ранее.

Оператор DATA SAVE DC имеет следующую структуру:

DATA SAVE DC [¤] [# <номер файла>,] { <список аргументов 1 > | END }

<**список аргументов**1> : := <аргумент 1> [{, <аргумент 1>}...]

<**аргумент** 1> ::= <а.в.>| <символьная переменная>| < символьная константа> | <метка массива>

Оператор DATA SAVE DC предназначен для вывода данных из списка аргументов 1 на диск в открытый файл в виде одной логической записи. Вывод выполняется в файл с указанным в операторе номером или (по умолчанию) в файл с номером 0.

Если файл не открыт, будет ошибка (ERR. 60).

Одна логическая запись выводится, начиная с текущего сектора (его номер система формирует и хранит в таблице устройств), и занимает столько секторов, сколько необходимо для размещения данных списка аргументов 1 вместе с разделителями в соответствии с форматом и правилами размещения данных в секторе.

В таблице устройств текущим сектором становится первый свободный в файле сектор.

Если указан параметр END, на диск выводится закрывающий блок файла, который занимает один сектор. Параметр ¤ означает запись с последующим контрольным чтением.

Пример 87. Запись правильного синтаксиса:

```
10 DATA SAVE DC A()
20 DATA SAVE DC #1,M%(),K%()
30 DATA SAVE DC %#2,B(),"УРОВЕНЬ 1"
40 DATA SAVE DC #1,END
```

Размещение данных в секторе и расчет резерва секторов для файла. Порция вывода — логическая запись, включающая совокупность данных из списка аргументов 1 операторов DATA SAVE DC и DATA SAVE DA вместе с разделителями. Она занимает на диске один или несколько (целое число) секторов. Каждое данное сопровождается служебной информацией, которая в формате файла обозначена как разделитель эле-

ментов. Разделитель располагается перед данными и занимает 2 байт. Константа на диске занимает столько байтов, сколько в ней записано символов. Значение символьной переменной и элемента символьного массива занимает столько байтов, сколько составляет дх длина по объявлению; целое число занимает 2 байт; действительное число — 8 байт. Объем сектора для информации — 255 байт.

При выводе данное вместе с разделителем должно полностью размещаться в одном секторе (без деления). Если данное не умещается в секторе, то оно записывается в следующий сектор, а незаполненная данными часть текущего сектора до конца заполняется символом «пусто» (HEX (00)). Это следует учитывать при расчете резерва секторов для файла и при формировании списка аргументов 1, а так как от этого зависит эффективность использования места на диске.

Например, при выполнении DATA SAVE DC «А» один символ «А» составляет логическую запись и на диске для него нужен один сектор, хотя в этом секторе символ займет всего 3 байт (2 байт разделитель и 1 байт сам символ), а остальные 252 байт заполнены разделителем (2 байт) и символами «пусто» (250 байт).

Пример 88.

1. Логическую запись составляет действительный массив А, имеющий 80 элементов. Один элемент с разделителем занимает на диске 10 байт.

В одном секторе можно разместить 25 чисел, следовательно, для логической записи нужен разерв в три сектора (в третьем секторе записи занято всего 50 байт).

2. Логическую запись составляет целый массив В %, содержащий 100 элементов, символьная переменная С¤ длиной 157 байт и десятиэлементный действительный массив К.

Если оператор записать в виде DATA SAVE DC В % ( ), С¤, К ( ), то для логической записи нужно резервировать четыре сектора: 1-й сектор для элементов В %, 2-й — для 37 элементов В %, 3-й для С¤, 4-й — для массива К.

Если оператор записать в виде DATA SAVE DC В % ( ), К ( ), С¤, то для логической записи нужно резервировать три сектора, так как массив К размещается во 2-м секторе логической записи, а для символьной переменной места недостаточно. Итак, в 1-м секторе размещаются 63 элемента массива В %, во 2-м — оставшиеся 37 элементов массива В % и массив К, в 3-м — С¤.

Для расчета резерва секторов файла необходимо рассчитать резерв каждой логической записи вывода и к сумме резервов для логических записей добавить единицу (для закрывающего блока). Полученное значение указывается в операторе DATA SAVE DC OPEN.

Оператор DATA LOAD DC. Оператор имеет следующую структуру:

DATA LOAD DC [# <номер файла>,] <список аргументов>
<**список аргументов**> : := <аргумент>[{, <аргумент>} ...]
<**аргумент**> : := <переменная> | <метка массива>

Оператор DATA LOAD DC предназначен для считывания значений логических записей из открытого файла, начиная с текущей логической записи, и присвоения считанных значений переменным и массивам из списка аргументов. Список аргументов определяет порцию ввода. Если данных в одной логической записи файла недостаточно для за-

полнения списка аргументов, то считываются последующие логические записи до окончательного заполнения списка аргументов (порция считывания больше порции вывода).

Если логическая запись полностью не использована в порции ввода, т. е. в списке аргументов нет места для считанных значений, то логическая запись считывается до конца, данные остаются неиспользованными, текущей становится следующая логическая запись (номером текущего сектора в таблице устройств становится номер первого сектора следующей логической записи). Попытка считывания информации из неоткрытого файла приводит к ошибке (ERR 60), а считывания за пределами резерва файла — к ошибке (ERR 76).

Итак, реализация оператора заканчивается, если исчерпан (заполнен) весь список аргументов.

Пример. 89. Запись правильного синтаксиса:

```
10 DATA LOAD DC A,C%(),K%
20 DATA LOAD DC #5,L%()
```

Если считан закрывающий блок файла, то запись не выполняется, считывание прекращается и номер текущего сектора не меняется.

Перемещения в файле операторами DSKIP, DBACK SPACE. Язык БЕЙСИК позволяет сделать текущей любую логическую запись или любой сектор в открытом файле, т. е. как бы переместиться в файле вперед к концу файла и назад к началу файла на заданное число секторов или логических записей. Выход за пределы файла невозможен. Перемещение условное, так как операторы выполняют изменение номера текущего сектора в строке таблицы устройств с номером, равным номеру файла.

Структура оператора DSKIP имеет вид:

DSKIP [# ⟨номер файла⟩,] { ⟨а. в.⟩ [S] | END }

Оператор DSKIP предназначен для пропуска секторов, если задан параметр S, или логических записей.

Число секторов и логических записей задается значением арифметического выражения без дробной части и знака.

Если указан параметр END, то текущей становится последняя логическая запись, занимающая конечный сектор файла, закрывающий блок файла. Конечный сектор становится также текущим, если значение арифметического выражения больше фактического числа секторов (логических записей).

Пример 90.

```
10 DSKIP END:REM   -ПЕРЕМЕЩЕНИЕ К КОНЦУ ФАЙЛА;
20 DSKIP #2,2:REM  -ПЕРЕМЕЩЕНИЕ НА 2 ЛОГИЧЕСКИЕ ЗАПИСИ;
30 DSKIP #1,3S:REM -ПЕРЕМЕЩЕНИЕ НА 3 СЕКТОРА;
```

Структура оператора DBACKSPACE имеет вид:

DBACKSPACE [# ⟨номер файла⟩,] { ⟨а. в⟩ [S] / BEG }

Оператор DBACKSPACE предназначен для возврата назад на заданное арифметическим выражением число секторов, если задан параметр S, или логических записей, если S не указан.

Параметр BEG означает возврат к началу файла. Он задает текущей записью первую логическую запись файла, т. е. в строке таблицы устройств с номером, равным номеру файла, значением текущего сектора становится начальный сектор файла.

Пример 91.

```
10 DBACKSPACE BEG
20 DBACKSPACE #1,55
30 DBACKSPACE #2,3
```

Ликвидация файла. Оператор SCRATCH. Структура оператора:

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

⟨список имен⟩ : : = ⟨имя⟩ [{, ⟨имя⟩}...]

Оператор SCRATCH предназначен для присвоения файлам зоны каталога, перечисленным в списке имен, статуса «ликвидируемый».

«Ликвидируемый» файл не считывается и не копируется, т. е. к нему нет доступа с помощью операторов каталога. В справке, получаемой с диска оператором LIST DC, этот файл помечен буквой S в графе TYPE (SD или SP). Секторы файла могут быть использованы для записи нового файла. Возможность доступа к ликвидируемому файлу данных можно восстановить, если файл открыть оператором DATA SAVE DC OPEN с прежним именем (признак S исчезает). Программный файл не восстанавливается.

Пример 92.

```
10 SCRATCH F"СПРАВ 1","СПРАВ 2","ИСХОД"
20 DATA SAVE DC OPEN F("СПРАВ 1")"ИТОГ"
30 SCRATCH F,1C,"ИТОГ"
40 SELECT #01CF
50 DATA SAVE DC OPEN F("ИТОГ")"ИТОГ"
```

В первом случае восстановился доступ к файлу данных с новым именем, во втором случае - с прежним именем.

Копирование информации с диска на диск. Операторы MOVE, COPY. Копирование информации с диска на диск можно выполнить в двух режимах:

1) копирование информации с каталогизированного диска на каталогизированный диск по файлам оператором MOVE;

2) физическое копирование информации (сектор в сектор), не зависящее от структуры информации на диске, оператором COPY.

Если при копировании информаци нет необходимости сохранить на диске для копии часть прежней информации, т. е. следует считать его «чистым», то перед копированием диск необходимо форматизовать, а для копирования по файлам и каталогизировать, задавая размеры зон каталога и указателя каталога в соответствии с объемом зон исходного диска.

Операторы копирования используют два устройства: исходное (откуда считывается информация) и для копии (куда записывается копия). В операторах указывается лишь один параметр <устройство> для задания устройства для копии. Если параметр не указан, то диск для копии установлен на то же устройство, где установлен исходный диск. Исходное устройство всегда задается оператором SELECT DISK, который должен выполняться до копирования. Устройство с ФАУ 18 можно не задавать.

При копировании тип исходного диска и тип диска копии должны быть различны, при этом в операторах копирования слева указывается тип исходного диска, а справа — тип диска копии:

FR (с F на R) или RF (с R на F).

Записи FF и RR недопустимы.

Проверить правильность копирования можно оператором VERIFY.

Структура оператора MOVE имеет следующий вид:

MOVE [⟨устройство⟩,] $\left\{\begin{matrix}\text{FR}\\\text{RF}\end{matrix}\right\}$

Оператор MOVE предназначен для копирования каталога диска на другой диск с удалением всех ликвидируемых файлов из зоны каталога и их имен из зоны указателя каталога. Информация о копируемых файлах при записи в УКФ диска копии редактируется в соответствии с фактическим местоположением файла на диске копии.

Вре́менные рабочие файлы, расположенные на исходном диске за зоной каталога, не копируются. Если перед копированием на диске копии уже была информация, копируемые файлы записываются после нее; при этом попытка скопировать файл с именем файла, уже имеющегося на диске копии, приводит к ошибке.

Пример 93.

```
10 MOVE FR:REM     -КОПИРОВАНИЕ С 18F НА 18R
20 SELECT DISK1CR
30 MOVE RF:REM     -КОПИРОВАНИЕ С 1CR НА 1CF
40 MOVE /18,FR:REM-КОПИРОВАНИЕ С 1CF НА 18R
```

Структура оператора COPY имеет следующий вид:

COPY [⟨устройство⟩,] $\left\{\begin{matrix}FR\\RF\end{matrix}\right\}$(⟨а.в.1⟩, ⟨а.в.2⟩)

Оператор COPY предназначен для физического копирования с диска на диск. Копирование выполняется в указанном диапазоне секторов. Параметр <а.в. 1> задает адрес первого сектора, параметр <а. в. 2> — адрес последнего сектора копируемой области. Информация при копировании не анализируется, не изменяется и не перемещается, т. е. информация с пятого сектора копируется строго в пятый, из десятого в десятый и т. д.

Пример 94.

```
10 COPY FR(0,1000):REM   -ПОЛНОЕ КОПИРОВАНИЕ ГИБКОГО ДИСКА
20 COPY #1C,RF(110,300):REM-УКАЗАННАЯ ЧАСТЬ ДИСКА КОПИРУ
                           ЕТСЯ С 10R НА 1CF
30 SELECT DISK1CF
40 COPY #18,FR(30,100):REM -УКАЗАННАЯ ЧАСТЬ ДИСКА КОПИРУ
                           ЕТСЯ С 1CF Н А 10R
```

Формирование файла данных. Ввод данных из файла. Все сказанное ранее о файлах и способах работы с файлами данных можно использовать в различных ситуациях, возникающих при вводе — выводе.

Рассмотрим схематически программную реализацию ввода-вывода данных в файл в зависимости от наличия файла на диске, уровня заполняемости и т. д.

Вывод данных в файл. Если файла нет на диске, его нужно открыть в свободной части каталога или на месте прежнего файла, подсчитав предварительно резерв секторов для файла, который либо указывается при открытии файла в DATA SAVE DC OPEN, либо используется для оценки возможности размещения нового файла на месте прежнего.

Прежний файл необходимо ликвидировать.

Далее в открытый файл можно выводить данные одной или несколькими логическими записями (порциями вывода). При этом вывод каждой логической записи выполняется оператором DATA SAVE DC. Логические записи по структуре могут быть разными или одинаковыми.

Для вывода одинаковых логических записей можно организовать цикл. За последней логической записью необходимо вывести закрывающий (конечный) блок файла тем же оператором с параметром END, после чего файл можно закрывать.

Схематически последовательность действий и используемых при программировании операторов можно представить так:

подсчитать резерв секторов файла;

SCRATCH, если новый файл открывается на месте прежнего;

выбрать номер для открываемого файла;

SELECT #;

DATA SAVE DC OPEN;

DATA SAVE DC... логическая запись — повторяется многократно, т. е. записывается столько раз, сколько исполняется или используется в цикле;

DATA SAVE DC END;

DATA SAVE DC CLOSE — не обязательно.

Если файл есть на диске, его нужно открывать как существующий. Затем необходимо установить текущую логическую запись для вывода, так как после открытия файла текущей записью является первая логическая запись файла. Чаще всего в такой ситуации выполняется дозапись в конец файла, т. е. на место записанного ранее закрывающего блока файла.

Итак, схема такова:

выбрать номер для открываемого файла;

SELECT #;

DATA LOAD DC OPEN;

DSKIP END;

DATA SAVE DC — повторяется многократно;

DATA SAVE DC END;

DATA SAVE DC CLOSE — не обязательно.

Ввод данных из файла. Вводить данные можно из файла, открытого ранее (например, для вывода), в простивном случае файл нужно открывать как существующий. Если файл был открыт, то необходимо установить текущую запись для чтения (DBACK SPACE —с начала файла). Если файл открывается заново, текущей становится первая логическая запись файла.

Последовательность действия:

выбрать номер файла (для чтения);

SELECT #;

DATA LOAD DC OPEN, если ранее файл не открывался;

DSKIP или DBACKSPACE — для установления текущей записи;

DATA LOAD DC;

IF END THEN — анализ на конец файла при последовательном считывании в цикле.

Считывание можно организовать по счетчику (в цикле), тогда оператор IF END THEN можно не использовать. Можно также закончить считывание и по признаку, введенному из файла.

Если один и тот же файл приходится использовать и для записи (дозаписи) и для чтения, не следует забывать о том, что для них необходимо порознь учитывать состояние текущей логической записи. Такую задачу можно поручить системе. Для этого файл нужно открыть дважды: сначала для записи (DATA SAVE DC OPEN), а затем с другим номером — для чтения (DATA LOAD DC OPEN). Тогда состояние файла для вывода и состояние файла для чтения будет учтено в разных строках таблицы устройств.

Аналогично можно открывать файл для записи и дозаписи (для редактирования).

Пример 95.

Необходимо сформировать на диске файл, логическими записями которого будут строки документа типа «справочник». Реквизиты строки: наименование, код, цена. Всего не более 100 строк. Предусмотреть дозапись в файл и распечатку справочника.

```
10 DIM N$(100),K%(100),C(100)
20 REM ФОРМИРОВАНИЕ СПРАВОЧНИКА
30 INPUT "ЗАДАЙТЕ РЕЖИМ ФОРМИРОВАНИЯ:1-ЗАПИСЬ,2-ДОЗАПИСЬ,3-Р
АСПЕЧАТКА,4-КОНЕЦ",P%
40 ON P%GOTO70,110,230,300
50 PRINT /05,"ОШИБКА В ЗАДАНИИ РЕЖИМА",HEX(07)
60 GOTO 30
70 REM ЗАПИСЬ
80 SELECT #118
90 DATA SAVE DC OPEN F#1,(102)"ПРОД-Я"
100 GOTO 150
110 REM ДОЗАПИСЬ
120 SELECT #118
130 DATA LOAD DC OPEN F#1,"ПРОД-Я"
140 DSKIP #1,END
150 INPUT "ВВЕДИТЕ РЕКВИЗИТЫ СТРОКИ: НАИМЕНОВАНИЕ ИЛИ ***",N
$:REM ВВОД ОЧЕРЕДНОЙ СТРОКИ С УК
160 IF N$="***"THEN200
170 INPUT "КОД,ЦЕНА",K%,C
180 DATA SAVE DC #1,N$,K%,C
190 GOTO 150
200 DATA SAVE DC #1,END
210 DATA SAVE DC CLOSE
220 GOTO 30
230 REM РАСПЕЧАТКА СПРАВОЧНИКА
240 SELECT #018:PRINT "СПРАВОЧНИК ПРОДУКЦИИ"
245 N=1
250 DATA LOAD DC OPEN F"ПРОД-Я"
260 DATA LOAD DC A$,B%,C:IF ENDTHEN 30
270 PRINTUSING 280,N,A$,B%,C:N=N+1
280 %###  ################  ####  ####.##
290 GOTO 260
300 END
```

Получение справочной информации о файле в программе оператором LIMITS. Оператор LIST DC выдает справочную информацию о файле для визуального контроля и анализа. Аналогичную информацию можно получить в числовых переменных, чтобы выполнить анализ состояния файла в программе. Такую информацию выдает оператор

LIMITS. Он записывает в числовые переменные адрес начального сектора в файле, адрес конечного сектора файла и число секторов, фактически занятых файлом (1-я форма оператора), или номер текущего сектора (2-я форма оператора). Итак, оператор LIMITS имеет две формы:

1) **LIMITS** <тип диска> [#<номер файла>], <имя>, <числовая переменная 1>, <числовая переменная 2>, <числовая переменная 3>
2) **LIMITS** <тип диска> # <номер файла>, <числовая переменная 1>, <числовая переменная 2>, <числовая переменная 3>

Оператор LIMITS предназначен для получения пользователем информации о местоположении файла и числе секторов, занимаемых файлом на диске.

В операторе LIMITS (форма 1) задается параметр <имя> — данные о файле с заданным именем из указателя каталога переписываются в переменные, при этом значения параметров <числовая переменная> с номерами 1, 2, 3, становятся равными соответственно начальному адресу сектора в файле, конечному адресу сектора в файле и числу используемых секторов в файле.

Для реализации оператора LIMITS (форма 1) необходимо, чтобы файл имел закрывающий блок (сформированный при выполнении оператора DATA SAVE DC END), так как только при его наличии определяется число секторов в файле в указателе каталога.

При реализации оператора LIMITS с параметром <имя> перепись информации из указателя каталога в переменные идет через таблицу устройств (номер строки таблицы устройств, через которую идет перепись информации, задается параметром <номер файла>, если он не задан, то через строку 0).

Таким образом, параметры файла в этой строке таблицы будут искажены. Поэтому при выполнении данного оператора не рекомендуется перепись через строку таблицы устройств, приписанную открытому файлу, так как иначе информация о нем будет утеряна и следующие обращения к нему (особенно по оператору DATA SAVE DC) могут привести к искажению информации в файле.

В операторе LIMITS (форма 2) обращение к файлу идет по номеру файла так, как к открытому файлу. При реализации оператора LIMITS переменным присваиваются значения начального, конечного и текущего адресов сектора открытого файла из таблицы устройств. Номер строки таблицы устройств соответствует значению параметра <номер файла>; по умолчанию рассматривается строка 0. При реализации этой формы оператора LIMITS обращения к диску нет.

Пример 96.

```
10 LIMITS F"ДАННЫЕ",A,B,C
20 LIMITS T#1,"ЗАЧА",X,Y,M(1)
30 LIMITS T#1,X(1,1),X(1,2),X(1,3)
```

Ввод-вывод программы на диск. Операторы LOAD DC, SAVE DC. Структура оператора LOAD DC:

LOAD DC ⟨тип диска⟩ [⟨устройство⟩,] ⟨имя⟩ [⟨номер строки 1⟩]. [,⟨номер строки 2⟩]

Оператор LOAD DC предназначен для загрузки программы (или ее сегмента) с заданным именем, записанной на диске, в память машины.

Если оператор LOAD DC используется при счете по программе, то при его выполнении происходит следующее:

1) приостанавливается исполнение текущей программы;

2) из памяти машины стираются все строки текущей программы или ее сегмента, ограниченного номерами строк. Если не указан один из параметров, то подразумевается соответственно номер начальной или конечной строки программы;

3) стираются значения всех переменных, кроме общих (объявленных в операторе COM);

4) программа с данным именем загружается в память машины с выбранного диска. Если программа с данным именем отсутствует, выдается сообщение об ошибке (ERR 73);

5) программа запускается на счет с номера строки, указанного параметром <номер строки 1> в операторе LOAD DC, а при его отсутствии с начала программы:

6) если номера строк не заданы, то стирается программа целиком, счет начинается с первого оператора загружаемой программы.

При отсутствии в загружаемой программе номера строки, указанного первым параметром <номер строки>, машина выдает сообщение об ошибке (ERR 11), если оператор был использован при счете по программе.

Оператор LOAD DC при использовании его в счете по программе позволяет организовывать автоматическое прохождение сегментированных программ. При этом для передачи значений переменных из одного сегмента программы в другой переменные должны быть объявлены как общие.

Оператор LOAD DC должен быть последним исполняемым оператором в строке.

Адрес с диска, с которого загружается программа, задается параметрами <тип диска> |<устройство>|.

Изменение оператора LOAD DC с указанием номеров строк в непосредственном счете отличается от его исполнения при счете по программе тем, что выполняются лишь действия по п. 2 и 4, а LOAD DC без указания строк лишь загружает программу.

Пример 97. Запись правильного синтаксиса:

```
10 LOAD DC R"ПРОГР1"
20 LOAD DC F#2,"РЯД"250,1000
30 X=5:LOAD DC R#X,"ИЯ"200
40 LOAD DC T"ИМЯ 1",500
```

Структура оператора SAVE DC:

SAVE DC ⟨тип диска⟩ [¤][⟨устройство⟩,] [⟨признак⟩] $\left[\left(\left\{\begin{array}{l}<\text{а. в.}>\\<\text{имя 1}>\end{array}\right\}\right)\right]$ ⟨имя⟩

[⟨номер строки 1⟩] [,⟨номер строки 2⟩]

Здесь <признак> : : =P|T|G; <имя 1> — имя ликвидируемого файла; <имя> — имя записываемой программы.

Оператор SAVE DC предназначен для записи программ (сегментов программ) на заданный диск. В указателе каталога записываются имя файла, тип файла (программа), начальный адрес и конечный адрес файла (номера начального сектора и конечного сектора файла).

Программа автоматически записывается в секторах, отведенных машиной под файл.

Параметр ¤ задает контрольное считывание после записи. Эта проверка увеличивает время на реализацию оператора SAVE DC.

Параметр <а. в.> задает резерв секторов сверх необходимого числа секторов, занимаемых программой. Дополнительные секторы можно использовать для последующего расширения программы. Чтобы записать программу на место ликвидируемого файла программы или данных, отмеченного оператором SCRATCH, задается параметр <имя 1>. Если числа секторов, занятых ликвидируемым файлом, не достаточно для записи нового программного файла, то выдается сообщение об ошибке (ERR 76).

Параметры <имя 1> и <имя> могут совпадать, но <имя> не должно совпадать с именами других файлов, занесенных в каталог.

Если в операторе не задан ни параметр <имя 1>, ни параметр <а. в.>, то новая программа записывается в свободной области каталога и ей отводится столько секторов, сколько для нее требуется.

Параметры P, T и G обозначают признак защиты программы при записи на диск:

P —запись нетранслированной программы с защитой,

T — запись транслированной программы без защиты,

G — запись транслированной программы с защитой.

Параметры P и G указывают на то, что программа защищена от распечатки и повторной записи, т. е. она может быть лишь загружена и выполнена. Перед записью или распечаткой любой программы после загрузки защищенной программы необходимо полностью очистить память по оператору CLEAR.

Параметры < номер строки 1> и <номер строки 2> определяют первую и последнюю строки сегмента записываемой программы.

3. ОБМЕН ИНФОРМАЦИЕЙ С ДИСКОВЫМИ УСТРОЙСТВАМИ В РЕЖИМЕ АДРЕСАЦИИ СЕКТОРОВ

В режиме адресации секторов разрешается доступ к секторам диска операторами режима. Эти операторы отличаются от операторов каталога наличием буквы A в имени операторов вместо буквы C

и структурой параметров. Ввод-вывод данных выполняется либо логическими записями (в операторе DA), либо секторами — физическими записями (в операторе BA). Размер физической записи составляет 256 байт.

Возможно обращение операторами режима адресации к файлам каталогизированного диска, при этом следует соблюдать особую осторожность.

Забота о хранении и обработке всей информации в режиме адресации ложится на пользователя.

Возможно использование таблицы устройств и обращения к устройству по логическому номеру. В операторах режима адресации параметр <адрес сектора> задается символьной переменной, символьной константой, арифметическим выражением. При символьном задании номер сектора — двоичное число, являющееся значением первых двух байтов.

Оператор DATA SAVE DA. Структура оператора:

DATA SAVE DA <тип диска> [¤] [<устройство>,] (< адрес сектора>, <переменная>) {END / <список аргументов 1>}

Оператор DATA SAVE DA предназначен для записи данных из списка аргументов 1, начиная с сектора, номер которого задан значением параметра <адрес сектора>. Данные записываются в том же формате, что и в режиме каталога по операторам DATA SAVE DC. Каждый оператор DATA SAVE DA записывает на диск одну логическую запись, занимающую один или несколько секторов диска.

После выполнения оператора DATA SAVE DA в параметр <переменная> заносится номер сектора, следующего за последним сектором, занятым данной логической записью. Если в качестве этого параметра задана числовая переменная, то заносится десятичный номер, если символьная — в ее первые 2 байта заносится номер в виде двоичного кода.

Наличие в операторе параметра ¤ означает контрольное чтение данных после записи.

Если в операторе задан параметр END, то на диск записывается логическая запись конца файла, которую можно использовать для поиска конца файла оператором IF END THEN.

Пример 98.

```
10 DATA SAVE DA F(20,B)X$,B
20 DATA SAVE DA F/1C,(C$,C%)5+10/X,"ПРИМЕР",A
30 DATA SAVE DA F#5,(A,B)END
40 DATA SAVE DA R/18,(A,B)END
```

Оператор DATA LOAD DA. Структура оператора:

DATA LOAD DA <тип диска> [<устройство>,] (<адрес сектора>, <переменная>) <список аргументов>

Оператор DATA LOAD DA предназначен для считывания логических записей с диска, начиная с сектора, номер которого задан параметром <адрес сектора> с последующим присвоением значений переменным (массивам) из списка аргументов.

Считывание информации возможно, если она записана по операторам DATA SAVE DA или DATA SAVE DC.

Если в одной логической записи недостаточно данных для заполнения списка аргументов, то считывается следующая логическая запись.

Если в логической записи данных больше, чем требуется для заполнения списка аргументов, то логическая запись считывается до конца, лишние данные игнорируются.

После выполнения оператора DATA LOAD DA в параметр <переменная> заносится номер сектора, с которого начинается следующая логическая запись. Если в качестве этого параметра использована числовая переменная, то заносится десятичный номер, если символьная — номер заносится в ее первые 2 байт в виде двоичного числа.

Если при считывании логической записи встретилась запись окончания файла, то считывание данных прекращается, а значение <переменная> устанавливается равным номеру сектора, в котором содержится эта запись, а не номеру сектора, следующего по порядку.

Пример 99.

```
10 DATA LOAD DA R(A%,C%)X,Y(),B
20 DATA LOAD DA F#1,(A,B)Z%()
30 DATA LOAD DA F/18,(B+5,L%)X(),Y(5,1)
```

Оператор DATA SAVE BA. Структура оператора:

DATA SAVE BA <тип диска> [¤] [<устройство>,] (<адрес сектора>, <переменная>) {<символьная переменная> | <метка символьного массива>}

Оператор DATA SAVE BA предназначен для записи символьной информации на диск, помещающейся в один сектор (256 байт).

Если символьный массив содержит более 256 байт, то на диск выводятся лишь первые 256 байт.

Если символьный массив (переменная) содержит менее 256 байт, то остаток сектора заполняется кодами HEX (00).

Данные записываются в сектор, номер которого задан параметром <адрес сектора>, а номер следующего за ним сектора заносится в параметр <переменная>. Если в качестве этого параметра задана числовая переменная, то в нее заносится десятичный номер, если символьная — то в ее первые 2 байт заносится номер в виде двоичного числа.

Наличие в операторе параметра ¤ означает контрольное чтение данных после записи.

Данные, записанные по оператору DATA SAVE BA, могут быть считаны лишь по оператору DATA LOAD BA.

Пример 100.

```
10 DATA SAVE BA R(A%,B%)C%()
20 DATA SAVE BA F/1C,(C,C)A%()
30 DATA SAVE BA F#5,(A%,A%)E%
40 DATA SAVE BA R/1C,(L,L)X%
```

Оператор DATA LOAD BA. Структура оператора:

DATA LOAD BA <тип диска> [<устройство>], (<адрес сектора>, <переменная>) {<символьная переменная> | <метка символьного массива>}

Оператор DATA LOAD BA предназначен для считывания информации из одного сектора на диске в символьную переменную или символьный массив.

При выполнении этого оператора считываются все 256 байт информации, содержащейся в секторе.

Информация считывается из сектора, номер которого задан параметром <адрес сектора>, номер следующего за ним сектора заносится в параметр <переменная>. Если в качестве этого параметра использована цифровая переменная, то в нее заносится десятичный номер сектора, если — символьная, то номер заносится в ее первые 2 байт в виде двоичного числа.

Если длина символьного массива больше 256 байт, то его остальные байты сохраняют старые значения.

Если длина символьного массива (переменной) недостаточна для размещения 256 байт, то последние считанные байты игнорируются.

Пример 101.

```
100 DATA LOAD BA F(L%,L%)A%()
110 DATA LOAD BA R#3,(20,L)A%
120 DATA LOAD BA F(20/X,L%)E%
```

Оператор SAVE DA. Структура оператора:

SAVE DA <тип диска> [¤] [<устройство>,] [<признак>] (<адрес сектора>, <переменная>) [<номер строки 1>] [, <номер строки 2>]
<**признак**> : := P|T|G

Оператор SAVE DA предназначен для записи программы (сегмента программы) в заданном месте диска.

Программа записывается на диске, начиная с сектора, номер которого задают параметром <адрес сектора>. Так как оператор SAVE DA не позволяет присваивать программе имя, то записанная с его по-

мощью программа может быть считана с диска лишь по оператору LOAD DA.

После выполнения оператора SAVE DA в параметр <переменная> заносится номер сектора, следующего за последним сектором, занятым записанной программой. Если этот параметр задан числовой переменной, то заносится десятичный номер, если символьной, то номер заносится в ее первые 2 байт в виде двоичного кода.

Параметр ¤ означает контрольное считывание после записи. Это увеличивает время выполнения оператора SAVE DA. Параметры P, T и G обозначают признак защиты программы при записи на диск.

Если защищенная программа была загружена в память машины, то любая другая программа может быть записана или распечатана только в том случае, когда перед ее загрузкой память машины была очищена оператором CLEAR или нажатием на кнопку SR.

Номера строк определяют первую и последнюю строки сегмента записываемой программы:

1) если задан только <номер строки 1>, то записывается сегмент программы, начиная со строки с данным номером и до конца;

2) если задан только <номер строки 2>, то записывается сегмент программы, начиная с ее первой строки и до строки программы с данным номером;

3) если номера строк не заданы, то записывается вся программа.

Использование оператора SAVE DA для записи программы в режиме каталога может привести к разрушению каталога.

Пример 102.

```
10 SAVE DA F#2,P(A1,A2)
20 SAVE DA R¤(C¤,C¤)300
30 SAVE DA F¤#3,(M+P,C),500
40 SAVE DA RP(A(10),A(K+1))10,30
50 SAVE DA T/1C,(90,C)
```

Оператор LOAD DA. Структура оператора:

LOAD DA <тип диска> [<устройство>,] (< адрес сектора>, <переменная>) [<номер строки 1>] [, <номер строки 2 >]

Оператор предназначен для загрузки с диска всей программы или части программы в память машины.

Параметр <адрес сектора> задает номер первого сектора диска, который содержит имя программы.

Оператор LOAD DA должен быть последним исполняемым оператором в строке.

Если оператор LOAD DA используется в счете по программе, то выполняется следующее:

1) приостанавливается исполнение текущей программы;

2) из памяти машины стираются все строки текущей программы или ее сегмента, ограниченного значениями параметров < номер строки>;

3) стираются значения всех переменных, кроме общих (т. е. объявленных в операторе COM);

4) программа или ее часть загружается в память машины;

5) возобновляется исполнение программы.

Исполнение оператора LOAD DA в непосредственном счете с параметрами <номер строки 1> и <номер строки 2> отличается тем, что выполняются лишь действия по п. 2 и 4, а LOAD DA без параметров <номер строки 1> и <номер строки 2> выполняет лишь загрузку программы. В операторе LOAD DA осуществляется стирание строк программы, хранящейся в памяти машины, по следующим правилам:

1) если заданы оба параметра <номер строки>, то стирается часть программы, ограниченная этими номерами строк; счет возобновляется со строки с номером, равным значению первого параметра <номер строки>;

2) если задан только первый параметр <номер строки>, то стирается часть программы, начиная с заданного номера строки и до конца; счет возобновляется с заданного в операторе номера строки;

3) если задан только второй параметр <номер строки>, то стирается часть программы от ее начала до заданного номера строки, счет возобновляется с первой строки загружаемой программы;

4) если параметры <номер строки> не заданы, то стирается вся программа; счет возобновляется с первого оператора загружаемой программы.

При отсутствии в загружаемой программе номера строки, указанного в первом параметре <номер строки>, машина выдает сообщение об ошибке (ERR 11).

Переменная, задаваемая в параметре <переменная>, должна быть объявлена как общая, иначе выдается сообщение об ошибке (ERR 22). В эту переменную после выполнения LOAD DA заносится номер сектора, следующего за последним сектором, занятым загружаемой программой. Если в качестве этого параметра используется цифровая переменная, то заносится десятичный номер, если — символьная, то номер заносится в ее первые 2 байт в виде двоичного числа.

Оператор LOAD DA в счете по программе позволяет пользователю организовать автоматическое прохождение сегментированных программ. Для передачи значений переменных из одного сегмента программы в другой переменные должны быть объявлены как общие.

Адрес диска, с которого загружается программа, задается параметрами <тип диска> [<устройство>].

Пример 103.

```
10 LOAD DA F(40,1)
20 LOAD DA F#2,(B%,C%)310,450
30 LOAD DA R#2,(2*5/X,X)300
40 LOAD DA F(A,A),400
```

Глава XII

МАТРИЧНЫЕ ОПЕРАЦИИ С МАССИВАМИ

1. ОБЩИЕ СВЕДЕНИЯ

Язык БЕЙСИК содержит группу операторов, называемых *матричными*, которые реализуют основные операции алгебры матриц: сложение матриц; вычитание матриц; умножение матриц; умножение матрицы на скаляр; получение матрицы, обратной исходной (инвертирование матрицы); транспонирование матрицы. Если эти операции выполнять без использования матричных операторов, потребуется больше машинного времени.

Матричные операторы позволяют вводить значения массивов, печатать массивы, присваивать массивам значения констант из DATA или значения другого массива (только для числовых массивов, а для символьных используется оператор MAT COPY), значения 0 и 1, а также значения единичной матрицы. Эти операторы можно назвать операторами, обрабатывающими массивы данных. Их можно использовать в любой задаче, а не только в математических задачах с применением операций над матрицами.

Матричные операторы синтаксически отличаются от остальных операторов языка, но используются они в программе наряду с другими.

Объектами обработки в матричных операторах являются только массивы [лишь в двух операторах MAT INV и MAT ( )$_*$ используется простая числовая переменная], имя операторов начинается с MAT, а массив обозначается не меткой, а идентификатором.

Массивы могут быть одномерными и двумерными. Во всех матричных операторах массивы числовые, а в MAT READ, MAT REDIM, MAT INPUT, MAT PRINT, кроме того, и символьные.

Итак, в матричных операторах обрабатываются те же массивы, что и в других (нематричных) операторах, только они обозначаются иначе и называются иначе (в соответствии с терминологией, принятой в алгебре матриц): двумерный массив называется матрицей, а одномерный — вектором.

Матричные операторы имеют еще одну особенность: только в матричных операторах переопределяется размерность массива. Переопределение выполняется автоматически операторами по правилам алгебры матриц (неявно). В некоторых операторах размерность массива переопределяется явным образом, т. е. указывается пользователем непосредственно в операторе (после идентификатора массива указывается число строк, число столбцов, а для символьных массивов еще и длина элемента, как в объявлении массива, только при переопределении используется арифметическое выражение).

Можно выделить общие правила использования матричных операторов.

1. Каждый идентификатор в матричном операторе обозначает одномерный или двумерный массив, кроме идентификатора переменной, обозначающей число при умножении матрицы на скаляр и идентифика-

тора переменной, в которой получается значение определителя исходной матрицы.

2. Каждый используемый в матричных операторах массив должен быть предварительно объявлен в операторах DIM и COM, т. е. для массива должен быть определен максимальный объем памяти, занимаемый массивом (в соответствии с объявленной размерностью). Необъявленный массив автоматически получает размерность 10·10. Одномерный массив может быть необъявленным, если ранее в программе было обращение к его элементу, такой массив автоматически получает размерность 10. Для необъявленных символьных массивов и для символьных массивов, у которых в объявлении не указана длина элемента, автоматически устанавливается длина элемента, равная 16 символам.

3. В операторах, разрешающих явное переопределение размерности массива, все параметры (число строк, число столбцов, длина символьного элемента) задаются арифметическим выражением, из которого затем используется целая часть. Значение целой части арифметического выражения, задающего длину элемента, не должно превышать 253, а остальных параметров — 7999 (как в DIM и COM); при этом массив после переопределения должен быть не больше объявленного в DIM или COM, т. е. число элементов числового массива и число символов символьного массива после переопределения не должны превышать соответствующих значений в объявлении массива.

4. Не допускается переопределение матрицы вектором (т. е. двумерного массива одномерным) и наоборот.

5. Значения числовых элементов при переопределении массива сохраняются, изменяются лишь длина строк, столбцов или общая длина массива; в символьных массивах неизменны лишь значения символов.

6. Для операторов, выполняющих неявное переопределение размерности массива, результирующий массив в DIM или COM следует объявлять так, чтобы соответствующие параметры (число строк и столбцов) в объявлении результирующего массива были не меньше тех значений, которые установятся при выполнении матричного оператора в соответствии с правилами алгебры матриц.

Не допускается многократное использование операций (по типу арифметического выражения) в одном матричном операторе, т. е. запись оператора MAT E = A — B + C синтаксически ошибочна.

Общие формы матричных операторов с указанием способа переопределения размерности массива приведены в табл. 11.

Для описания далее будут использоваться следующие конструкции синтаксиса матричных операторов:

<список массивов 1> : := <массив> [{,<массив>}...]

<массив> : := <числовой массив> | <символьный массив>

<список массивов> : :=<список числовых массивов>|<список символьных массивов>

<список числовых массивов>: :=<числовой массив> [{,< числовой массив>} ...]

<список символьных массивов>: :=<символьный массив> [{, <символьный массив>}...]

11. Матричные операторы

Оператор	Назначение оператора	Пример записи	Особенности выполнения
MAT REDIM	Переопределение размерности массива	MAT REDIM X (2, K)	*, **
MAT READ	Присвоение элементам массива значений констант оператора DATA	MAT READ A, B ¤	*, **
MAT =	Присвоение элементам числового массива значений элемента другого числового массива	MAT X = A	***
MAT ZER	Присвоение каждому элементу числового массива значения 0	MAT A = ZER	*
MAT CON	Присвоение каждому элементу числового массива значения 1	MAT A = CON	*
MAT IDN	Присвоение элементам числового массива значений элементов единичной матрицы	MAT A = IDN	*
MAT INPUT	Ввод значений элементов массивов	MAT INPUT A ¤, B %	*, **
MAT PRINT	Печать массивов	MAT PRINT A ¤, B; C	**
MAT +	Сложение матриц	MAT X = A + B	***
MAT −	Вычитание матриц	MAT X = A − B	***
MAT *	Умножение матриц	MAT X = A*B	***
MAT ( ) *	Умножение матрицы на скаляр	MAT X = (A)*K	***
MAT INV, d	Вычисление матрицы, обратной данной, и определителя	MAT X = INV (A), K	***
MAT TRN	Транспонирование матрицы	MAT X = TRN (A)	***

* — размерность массива переопределяется явно; ** — оператор применяется для числовых и символьных массивов; *** — размерность массива переопределяется неявно.

<**числовой массив**>: := <идентификатор переменной> [%] ([<(а.в.1> [, <а.в.2>])]

<**символьный массив**> : := <идентификатор переменной> ¤ [(<а.в.1> [,<а.в.2>])] [<а. в. 3>]]

Параметры <а. в. 1>, <а. в. 2>, <а. в. 3> — арифметические выражения, используемые лишь в операторах с явным переопределением размерности для обозначения новых значений размерности массива:

<а. в. 1> — числа элементов одномерного массива или числа строк двумерного массива;

<а.в.2> числа столбцов двумерного массива;

<а.в.3> — длина элемента символьного массива.

Оператор MAT REDIM рассмотрен выше.

2. ОПЕРАТОР MAT READ

Структура оператора:

MAT READ <список массивов 1>

Оператор MAT READ предназначен для присвоения элементам числовых и символьных массивов списка значений констант из оператора DATA.

Оператор MAT READ допускает явное переопределение размерности массивов. При выполнении оператора сначала для каждого массива из списка поочередно устанавливается новая размерность, если она указана арифметическими выражениями, а затем массив построчно заполняется значениями из DATA, начиная с текущей константы, т. е. с первой константы, неиспользованной ранее выполненным оператором READ или MAT READ, или с номера константы, установленной оператором RESTORE.

Если в операторах DATA недостаточно констант для заполнения всего списка массивов или тип констант не соответствует типу массива, то выдается сообщение об ошибке (ERR 27).

Пример 104.

```
10 DIM A(5),B$(2,3),C%(3,2),M$*6
20 READ M$:PRINT M$;
30 MAT READ A(3),B$(3,2):READ K$
40 MAT READ C%
50 PRINT "  ";K$
60 FOR N=1TO3
70 PRINT A(N),B$(N,1);B$(N,2);C%(N,1);C%(N,2)
80 NEXT N:END
90 DATA "ПРИМЕР",11,12,13,"C","B","A","1","2","3","MAT READ
",21,22,23,24,25,26

ПРИМЕР  MAT READ
 11              C               B               21  22
 12              A               1               23  24
 13              2               3               25  26
```

3. ОПЕРАТОР MAT =

Структура оператора:

MAT <числовой массив> = <числовой массив>

Оператор предназначен для присвоения элементам числового массива, указанного слева от символа « = », значений элементов числового массива, записанного справа от символа « = ».

Для успешного выполнения оператора необходимо, чтобы до выполнения его была установлена размерность левого числового массива не менее размерности исходного правого числового массива.

При выполнении оператора MAT= сначала размерность результирующего массива автоматически приводится в соответствие с размерностью исходного (правого) массива, а затем присваиваются значения.

Пример 105.

```
10 DIM A(1,2),B(1,3),C(1,3)
20 MAT READ A,B:PRINT A():PRINT B():PRINT C()
30 MAT C=B:MAT B=A:PRINT
40 PRINT A():PRINT B():PRINT C()
50 DATA 1,2,3,4,5

 1  2
 3  4  5
 0  0  0

 1  2
 1  2
 3  4  5
```

4. ОПЕРАТОРЫ MAT ZER, MAT CON, MAT IDN

Операторы **MAT ZER** и **MAT CON.** Они идентичны по конструкции и по назначению, допускают явное переопределение размерности. Структура операторов:

MAT <числовой массив> = **ZER** [(<а.в.1> [,< а.в.2 >])]
MAT <числовой массив> = **CON** [(<а.в. 1> [,<а.в.2 >])]

При выполнении операторов сначала переопределяется размерность числового массива, если новая размерность указана в операторе, а затем каждому элементу числового массива присваивается значение 0, если выполняется MAT ZER, и присваивается значение 1, если выполняется MAT CON.

Оператор MAT IDN. Структура оператора:

MAT <числовая матрица> = **IDN** [(<а.в. 1>[, <а.в.2>])]
<числовая матрица>: := <двумерный числовой массив>

Оператор предназначен для присвоения числовой матрице значения единичной матрицы.

По сравнению с операторами MAT ZER и MAT CON оператор MAT IDN имеет ограничения по размерности числового массива, продиктованные правилами алгебры матриц. Единичная матрица — квадратная матрица, элементы главной диагонали которой имеют значение 1, остальные элементы нулевые (у элементов главной диагонали номер строки равен номеру столбца).

Итак, числовой массив в операторе MAT IDN должен быть двумерным с одинаковым числом строк и столбцов, т. е. квадратной матрицей. Если массив объявлен прямоугольной матрицей, то его обязательно нужно переопределить явным образом в операторе.

При выполнении оператора сначала переопределяется размерность матрицы в соответствии с заданием в операторе, затем присваивается значение единичной матрицы. Если размерность не указана, переопределение не происходит.

Пример 106.

```
10 DIM A(4,4),B(4,4),C(4,4),L%(1)1
20 MAT A=ZER(1,16):MAT B=CON(3,3):MAT C=IDN(2,2)
30 MAT PRINT A;B;C;L%
40 MAT B=IDN:MAT A=CON
50 MAT PRINT B;A;L%

 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 1  1  1
 1  1  1
 1  1  1
 1  0
 0  1



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

5. ОПЕРАТОР MAT INPUT

Структура оператора имеет вид:

MAT INPUT [<устройство>,] <список массивов 1>

Оператор MAT INPUT предназначен для ввода с клавиатуры значений массивов списка. Оператор допускает переопределение размерности явным образом. Если в операторе не задан параметр <устройство>, то ввод выполняется с клавиатуры с ФАУ 01 или с устройства, заданного оператором SELECT CI.

При выполнении оператора MAT INPUT каждому массиву из списка сначала устанавливается новая размерность, если она была указана в операторе для этого массива, затем массив построчно заполняется значениями, набираемыми с клавиатуры. Если новая размерность не указана, переопределения не происходит.

При выполнении оператора MAT INPUT так же, как при выполнении INPUT, на экране индицируется символ «?» и машина переходит в

ожидание набора данных. Данные набираются как константы в соответствии с типом заполняемого массива.

Если символьное данное содержит символ «, », то символьное данное необходимо вводить в кавычках, в противном случае кавычки необязательны. Разделяются набираемые данные символом «, ». Завершается набор нажатием на клавишу CR/LF. Набор можно прервать, нажав дважды на клавишу CR/LF до полного заполнения массива значениями.

В массиве, ввод которого прерван, и в массивах, следующих за ним в списке, незаполненные элементы сохраняют прежние значения. После нажатия первый раз на клавишу CR/LF на экране вновь индицируется символ «?» и набор данных можно продолжить. Редактировать набираемые значения можно от символа «?» до текущего символа набора.

Если после нажатия на клавишу CR/LF машина обнаружила ошибку в наборе, то сообщается об ошибке и символ «?» индицируется вновь. Набор данных можно повторить, начиная с ошибочного. Набранные значения до ошибочного сохраняются.

Пример 107.

```
10 DIM A(5),B(1,5):MAT B=CON
20 MAT INPUT A(2),B:MAT PRINT B
```

RUN CR LF — запуск на счет
?1, 2, 3 CR LF — набранные значения присвоились элементам A(1), (A (2), B(1)
?4 CR LF — значение присвоилось B(2)
?CR LF — досрочное завершение набора
3 4 1 1 1 — в массиве B элементы после второго сохранили старые значения.

6. ОПЕРАТОР MAT PRINT

Оператор имеет следующую структуру:

MAT PRINT [<устройство>,] <идентификатор переменной> [{¤ / %}]
[{, / ;} <идентификатор переменной> [{¤ / %}]] ...]

Оператор MAT PRINT предназначен для вывода значений массивов, перечисленных в списке оператора, на устройство, заданное параметром <устройство>. По умолчанию вывод выполняется на БОСГИ с ФАУ 05 или на устройство, заданное оператором SELECT PRINT.

Вывод данных в массиве выполняется построчно в зонном формате, если за идентификатором в списке следует символ «,», или в плотном формате, если следует символ «;». Последний или единственный массив в списке выводится в зонном формате. Каждая строка каждого массива списка выводится с начала печатной строки. Чтобы последний массив списка печатался в плотном формате, в список можно добавить символ «;» и одноэлементный символьный массив, который распечатается пустой строкой.

Пример 108.

```
10 MAT PRINT USING "#",A%,B%,C
20 MAT PRINT X
100 DIM A$(1,1),B%(3,3),C(2),P$(1)1
110 A$(1,1)="PRINT "!MAT REDIM A$(1,5)1
120 MAT B%=IDN!MAT C=CON
130 MAT PRINT A$,B%,C,P$
```

```
P                R                I                N                T

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

7. ОПЕРАТОРЫ MAT+, MAT—

Операторы MAT + (матричное сложение) и MAT — (матричное вычитание) имеют идентичную структуру и правила выполнения:

MAT <числовой массив> <числовой массив>+<числовой массив>

MAT <числовой массив> = <числовой массив> — <числовой массив>

Оператор MAT + предназначен для сложения числовых матриц или векторов; оператор MAT — — для вычитания числовых матриц или векторов, указанных в правой части.

Для успешного выполнения операторов необходимо выполнить следующие требования:

массивы в правой части должны быть одного типа (действительные или целые);

массивы в правой части должны быть одновременно двумерными или одномерными с одинаковым числом строк и столбцов или элементов; допускается сочетание одномерного массива и одностолбцового двумерного массива, число строк которого равно числу элементов в одномерном массиве;

объявленная в операторах DIM или COM размерность числового массива, указанного в левой части рассматриваемых операторов, должна быть не меньше размерности массива в правой части (общие требования).

При выполнении операторов MAT + и MAT — для массива в левой части устанавливается автоматически результирующая размерность, соответствующая размерности массивов-слагаемых. Например, левый массив двумерный, а правые — одномерные массивы или одностолбцовые двумерные; тогда левый тоже становится одностолбцовым с

числом строк, равным числу элементов (или строк) массивов-слагаемых. После переопределения размерности элементам результирующего массива присваиваются значения суммы (разности) значений соответствующих элементов массивов-слагаемых.

Допускается использование одного и того же массива в качестве одного или двух массивов-слагаемых и в качестве результирующего массива.

Пример 109.

```
10 DIM A(1,5),B(1,5),C(1,5),P%(1)1
20 MAT READ A,B:MAT C=A+B:MAT R=B-A
30 MAT PRINT C;R;P%
40 DATA 1,2,3,4,5,11,12,13,14,15

 12  14  16  18  20
 10  10  10  10  10
```

Пример 110.

```
10 DIM A%(5,1),B%(5),P%(1)1
20 MAT A%=CON:MAT B%=CON
30 MAT A%=A%+B%:MAT REDIM A%(1,5)
40 MAT PRINT A%;P%

 2  2  2  2  2
```

Пример 111.

```
10 DIM A(5,5),B(2,3),C(2,3)
20 MAT READ B,C:MAT A=B-C
30 MAT PRINT A
40 DATA 12,23,36,45,56,69,1,1,3,1,1,3

 11              22              33
 44              55              66
```

8. ОПЕРАТОР MAT *

Оператор имеет следующую структуру:

MAT <числовой массив> = <числовой массив> * <числовой массив>

Оператор MAT* предназначен для умножения числовых массивов. Оператор реализуется в строгом соответствии с правилами матричной алгебры. Массивы правой части — сомножители, массиву в левой части присваивается результат матричного умножения.

Для успешной реализации оператора необходимо выполнить следующие требования:

массивы в правой части должны быть одного типа (действительные или целые);

число столбцов первого массива-сомножителя должно быть равно числу строк двумерного или числу элементов одномерного второго массива-сомножителя;

недопустимо умножение двух векторов, кроме одноэлементных.

При выполнении оператора результирующая размерность устанавливается по правилам: число строк равно числу строк первого массива-сомножителя, а число столбцов — числу столбцов второго массива-сомножителя. Значение элемента результирующего массива равно сумме попарных произведений значений элементов строки первого массива на значения элементов с теми же номерами из второго столбца массива. Номера строки и столбца, участвующих в данном вычислении, определяют местоположение элемента в результирующем массиве. Нельзя использовать массив в левой и правой частях одновременно.

Пример 112.

```
10 DIM A(4,2),B(2,2),C(4,4)
20 MAT READ A,B:MAT C=A*B:MAT PRINT C
30 DATA 5,1,0,2,1,1,3,2,1,0,1,1

 6             1
 2             2
 2             1
 5             2
```

Пример 113.

```
10 DIM A(3,2),B(2)
20 MAT READ A,B:MAT C=A*B:MAT PRINT C
30 DATA 0,1,4,5,3,2,1,1

 1
 9
 5
```

Пример 114.

```
10 DIM A(3),B(1,2)
20 MAT READ A,B:MAT C=A*B:MAT PRINT C
30 DATA 1,2,3,4,5

 4             5
 8             10
 12            15
```

9. ОПЕРАТОР MAT ( ) *

Структура оператора имеет следующий вид:

MAT <числовой массив> = (<а.в.>) * <числовой массив>

Оператор предназначен для скалярного умножения числа, заданного арифметическим выражением, и массива. Требования к массивам общие.

Результирующая размерность приводится в соответствие с размерностью числового массива-сомножителя. При выполнении оператора элементам результирующего массива присваиваются значения произведения числа на значение соответствующего элемента массива-сомножителя.

Пример 115.

```
10 MAT A=(16)*B
100 MAT X=(A)*B
200 MAT C%=(A+2.52)*B
300 DIM K(3,1),B(3):MAT B=CON
310 MAT K=(3)*B:MAT REDIM K(1,3)
320 MAT PRINT K

 3             3             3
```

10. ОПЕРАТОР MAT INV, d

Оператор имеет структуру:

MAT <числовой массив> = **INV** (<числовой массив>) [, <числовая переменная>]

Оператор предназначен для вычисления матрицы, обратной данной, и вычисления значения определителя. Размерность результирующего (левого) массива автоматически приводится к размерности исходного массива.

Требования, предъявляемые к массивам, задаются следующими правилами алгебры матриц:

1) исходная матрица должна быть квадратной, не сингулярной, т. е. имеющей обратную матрицу;

2) числовой массив для результата должен быть двумерным действительным массивом (матрицей) с размерностью, не меньшей размерности исходного массива.

Параметр <числовая переменная> указывается в операторе, если нужно получить значение определителя матрицы.

При выполнении оператора значение обратной матрицы заносится в результирующий числовой массив, а значение определителя присваивается числовой переменной.

Ввиду сложности вычислительной схемы реализации оператора MAT INV могут возникнуть ошибки вычислений, приводящие к переполнению или потере малых значений. Ликвидировать ошибку можно предварительным масштабированием исходной матрицы.

Полученная матрица может неточно совпадать с обратной за счет ошибок округления при расчетах для вычисления обратной матрицы.

Пример 116.

```
10 MAT X=INV(Y),C
20 MAT Z=INV(L)
100 MAT READ A(4,4)
110 MAT B=INV(A),L%
120 DATA 0,2,4,8,0,0,1,0,1,0,0,1,4,8,16,32
```

В результате выполнения программы матрица B получает размерность 4·4. Значение определителя равно 8.

11. ОПЕРАТОР MAT TRN

Оператор имеет следующую структуру:

MAT <числовой массив> = **TRN** (<числовой массив>)

Оператор предназначен для транспонирования числовой матрицы. Размерность результирующего массива автоматически приводится в соответствие с размерностью исходного массива.

Требования, предъявляемые к массивам:

1) числовые массивы должны иметь размерность в соответствии с правилами транспонирования;

2) массивы в левой и правой частях оператора не должны быть одномерными одновременно, если число элементов в них больше одного;

3) недопустимо использование одного массива в правой и в левой частях.

В результате выполнения оператора строки исходного массива становятся столбцами результирующего массива.

Пример 117.

```
10 DIM A(3,2)
20 MAT READ A
30 MAT B%=TRN(A):PRINT B%()
40 DATA 1,1,2,2,3,3
 1  2  3
 1  2  3
```

В результате выполнения программы матрица B % получает размерность 2·3.

Пример 118.

```
10 DIM A(1,2),B(10)
20 A(1,1)=2:A(1,2)=3
30 MAT B=TRN(A):PRINT B()

 2
 3
```

В результате выполнения программы вектор B получает размерность 2.

Пример 119.

```
10 DIM A(2)
20 A(1)=2:A(2)=3
30 MAT B=TRN(A):PRINT B()

 2  3
```

В результате выполнения программы матрица B получает размерность 1·2.

Глава XIII
ПРЕОБРАЗОВАНИЕ ИНФОРМАЦИИ

1. ОБЩИЕ СВЕДЕНИЯ

Язык БЕЙСИК позволяет выполнить операции с символьными данными. Среди них можно выделить следующие:

преобразование десятичного числа в однобайтное двоичное число и наоборот, выполняемые соответственно оператором BIN и функцией символьного переменного VAL;

вычисление значений числовых функций символьной переменной NOM, LEN, POS;

побитные преобразования, которые составляют логические операции, с помощью операторов AND, OR, XOR, BOOL, двоичное сложение, выполняемое оператором ADD, и циклический сдвиг двоичных разрядов в байте, выполняемый оператором ROTATE;

преобразование одного числа в символьный формат и наоборот, выполняемое оператором CONVERT;

преобразование нескольких чисел по формату, заданному пользователем, с одновременной упаковкой их в символьные переменные и обратное преобразование с распаковкой чисел, выполняемые операторами PACK и UNPACK;

преобразование кодов символов (побайтное преобразование значения символьной переменной или символьного массива), выполняемое оператором ¤ TRAN;

упаковка данных любого типа в символьную переменную или символьный массив по формату, заданному пользователем, и распаковка данных, выполняемые операторами ¤ PACK, ¤ UNPACK.

2. ПРЕОБРАЗОВАНИЕ ДЕСЯТИЧНОГО ЧИСЛА В ДВОИЧНОЕ И ФУНКЦИИ СИМВОЛЬНЫХ ПЕРЕМЕННЫХ

Функция символьной переменной LEN. Структура функции:

LEN (⟨символьная переменная⟩)

Функция LEN определяет длину символьной переменной без конечных пробелов, т. е. число значащих символов с начала переменной.

Если символьная переменная пуста, ее длина равна единице.

Пример 120.

```
10 Ax="ИСКРА 226"
20 PRINT LEN(Ax)
30 PRINT LEN(Bx)

 9
 1
```

Функция символьной переменной NUM. Структура функции:

NUM (⟨символьная переменная⟩)

Функция NUM определяет количество символов числа в символьной переменной. К символам числа относятся цифры от 0 до 9 включительно, точка, минус, плюс, пробел, E. Символьная переменная просматривается с начала до первого нечислового символа.

Пример 121.

```
10 Ax=" + 4849654 AA":Bx="KKK":Cx="-7A"
20 PRINT NUM(Ax);NUM(Bx);NUM(Cx)

 11  0  2
```

Функция символьной переменной POS. Структура функции:

POS (⟨символьная переменная⟩ ⟨операция сравнения⟩ { ⟨символьная переменная⟩ | ''⟨символ алфавита языка⟩'' | ⟨*hh*⟩ })

Функция POS определяет положение первого символа в символьной переменной, удовлетворяющего заданному условию. Каждый символ символьной переменной, начиная с первого, сравнивается с символом, заданным в кавычках HEX-кодом или первым символом второй символьной переменной. Сравнение выполняется до символа, удовлетворяющего условию, заданному операцией сравнения, или до конца переменной.

Если такого символа нет, то функция принимает значение 0.

Пример 122.

```
10 Ax="ММАКМКА":Bx="XXX"
20 PRINT POS(Ax="К");POS(Ax<44);POS(Ax=Bx)
 4  3  0
```

Функция символьной переменной VAL. Структура функции:

VAL ({ ⟨символьная константа⟩ })
{ ⟨символьная переменная⟩ }

Функция VAL преобразует двоичное значение первого символа (HEX-кода символа) символьной переменной или символьной константы в десятичное число (значение числа от 0 до 255).

Преобразование, выполняемое функцией VAL, обратно преобразованию, выполняемому оператором BIN.

Пример 123.

```
10 Ax="БЛОК":Bx="1-ВВОД"
20 PRINT VAL(Ax);VAL(Bx);VAL(Cx)
30 HEXPRINT STR(Ax,1,1);:HEXPRINT STR(Bx,1,1);STR(Cx,1,1)

 226  49  32
E23120
```

Для сравнения выведены HEX-коды, которым соответствуют двоичные значения 11100010 00110001 00100000

Оператор BIN. Структура оператора:

BIN (⟨символьная переменная⟩) = ⟨а. в.⟩

Оператор BIN предназначен для преобразования целой части арифметического выражения, значение которого должно быть не более 255, в двоичное восьмиразрядное число. Полученное восьмиразрядное число заносится в первый байт символьной переменной, т. е. становится кодом первого символа значения символьной переменной. Данное преобразование обратно преобразованию, выполняемому функцией VAL.

Пример 124.

```
10 BIN(A$)=16.25*4:PRINT A$
20 INIT ("A")B$:BIN(STR(B$,3,1))=233:BIN(B$)=233
30 PRINT B$

A
ИАИАААААААААААААА
```

3. ЛОГИЧЕСКИЕ ОПЕРАЦИИ И ДРУГИЕ ПРЕОБРАЗОВАНИЯ ДВОИЧНЫХ КОДОВ

Логические операции позволяют изменить один или несколько битов в байте символьного данного. При выполнении логических операций учитывается полная длина символьной переменной, включая и конечные пробелы. Операторы ADD, AND, OR, XOR, BOOL, ROTATE производят логические операции двух типов в зависимости от того, является ли второй параметр оператора двузначным шестнадцатеричным числом (HEX-кодом) или символьной переменной.

Параметры операторов называются логическими аргументами:

⟨**логические аргументы**⟩: : = (⟨символьная переменная⟩, (⟨*hh*⟩ | ⟨символьная переменная⟩))

В логической операции первого типа HEX-код участвует в операции с каждым байтом (HEX-кодом) первого логического аргумента.

В логической операции второго типа сначала выравниваются по длине значения логических аргументов, а затем выполняется операция с соответствующими парами байтов, принадлежащих разным логическим аргументам. Выравнивание длин логических аргументов выполняется по правилу чисел, т. е. недостающие позиции значения короткой символьной переменной дополняются слева нулями (HEX-кодами 00). (выравнивание происходит по правому краю).

Результат любой логической операции (кроме BOOL5) присваивается символьной переменной — первому логическому аргументу; если результат длиннее, то запоминается допустимая часть младших байтов результата.

Оператор ADD. Структура оператора:

ADD [C] ⟨логические аргументы⟩

Оператор ADD предназначен для сложения логических аргументов. Сложение выполняется в двоичной системе счисления.

Параметр C в операторе ADD не является обязательным. Если он не используется, то шестнадцатеричные цифры добавляются к каждому байту первого аргумента (сложение типа 1) или второй аргумент скла-

дывается с первым посимвольно (сложение типа 2); перенос между байтами отсутствует. Если параметр C включен в оператор, то прибавляемые байты складываются с первым аргументом так, как если бы он являлся одним символом, т. е. складываются два двоичных числа, и перенос между складываемыми байтами учитывается.

Пример 125.

Сложение типа 1, параметр C в операторе ADD не указан

```
5 DIM A$*2
10 A$=HEX$(01A3)
20 ADD (A$,82):HEXPRINT A$

8325
```

При выполнении программы производятся действия:

```
A¤ = 0000 0001 1010 0011   (01A3) — первый аргумент
     1000 0010 1000 0010   (8282) — второй аргумент
     ---------------------
     1000 0011 0010 0101   (8325) — результат сложения
```

и получается результат: A¤ = HEX (8325). Шестнадцатеричные цифры добавляются к каждому байту первой символьной переменной; перенос между байтами отсутствует, так как в операторе ADD нет параметра C.

Пример 126.

Сложение типа 1, параметр C указан в операторе ADD

```
5 DIM A$*2
10 A$=HEX$(01A3)
20 ADD C (A$,82):HEXPRINT A$

0225
```

При выполнении программы производятся действия:

```
A¤ = 0000 0001 1010 0011   (01A3)
 82          1000 0010   (  82)
     ---------------------
     0000 0010 0010 0101   (0225)
```

и получается результат: A¤ — HEX (0225). Складываются два двоичных числа: перенос между байтами существует, так как в оператор ADD включен параметр C.

Пример 127.

Сложение типа 2, параметр C в операторе не указан

```
5 DIM A$*3,B$*1
10 A$=HEX$(012345):B$=HEX$(FF)
20 ADD (B$,STR$(A$,2,2)):HEXPRINT B$
44
```

При выполнении программы производятся действия:

```
          B¤ = 0000 0000 1111 1111   (00FF)
STR(A¤, 2, 2) = 0010 0011 0100 0101   (2345)
                ---------------------
                0010 0011 0100 0100   (2344)
```

и получается результат: B¤ HEX (44). Перенос между байтами отсутствует, так как в оператор ADD не включен параметр C.

Складывается символьная переменная B¤ с частью символьной переменной A¤. Переменные имеют разную длину, недостающие позиции переменной B¤ заполняются нулями. Результат присваивается первой переменной B¤, длина которой меньше полученного результата, поэтому запоминается допустимая часть младших байтов результата — HEX (44).

Оператор AND. Структура оператора:

AND (логические аргументы)

Оператор AND выполняет операцию логического умножения (логическое И). Побитно сравнивая два аргумента, оператор присваивает значение 1 результирующему биту, если соответствующие биты сравниваемых величин равны 1, в противном случае присваивает значение 0.

Результат присваивается первой переменной. AND соответствует операции BOOL 8.

Пример 128.

```
5 DIM A$*1,B$*1
15 A$=HEX$(0C):B$=HEX$(08)
25 AND(A$,B$):HEXPRINT A$

08
```

При выполнении программы производятся действия:

```
A¤ = 0000 1100 (0C)
A¤ 0000 1000 (08)
   ---------
   0000 1000
```

и получается результат: A¤ HEX (08);

Пример 129.

```
5 DIM A$*2
1510 A$=HEX$(25A4)
1520 AND(A$,F0):HEXPRINT A$

20A0
```

При выполнении программы производятся действия:

```
A¤ = 0010  0101  1010  0100   (25A4)
     1111  0000  1111  0000   (F0F0)
     ---------------------
     0010  0000  1010  0000
```

и получается результат: A¤ = HEX (20A0)

Оператор OR. Структура оператора:

OR ⟨логические аргументы⟩

Оператор OR выполняет операцию логического ИЛИ. Побитно сравнивая два аргумента, оператор присваивает значение 0 результирующему биту, если соответствующие биты аргументов имеют нулевое значение, в противном случае присваивает значение 1.

Результат присваивается первой переменной. OR соответствует операции BOOLE.

Пример 130.

```
5 DIM A%1,B%1
15 A%=HEX(0C):B%=HEX(08)
25 OR(A%,B%):HEXPRINT A%

0C
```

При выполнении программы производятся действия:

```
A¤ = 0000  1100   (0C)
B¤ = 0000  1000   (08)
     ---------
     0000  1100
```

и получается результат: A¤ = HEX (0C).

Пример 131.

```
5 DIM A%2
10 A%=HEX(2544)
20 OR(A%,F0):HEXPRINT A%

F5F4
```

При выполнении программы производятся действия:

```
A¤ = 0010  0101  1010  0100   (25A4)
     1111  0000  1111  0000   (F0F0)
     ---------------------
     1111  0101  1111  0100
      F5          F4
```

и получается результат: A¤ = HEX (F5F4).

Оператор XOR. Структура оператора:

XOR ⟨логические аргументы⟩

Оператор XOR выполняет операцию исключающего ИЛИ. Побитно сравнивая два аргумента, оператор присваивает значение 0 результирующему биту, если соответствующие биты совпадают, и значение 1, если соответствующие биты разные. Результат присваивается первой переменной.

Пример 132.

```
5 DIM A%1,B%1
10 A%=HEX(0C):B%=HEX(08)
20 XOR(A%,B%):HEXPRINT A%

04
```

При выполнении программы производятся действия:

```
A¤ = 0000  1100   (0C)
B¤ = 0000  1000   (08)
     ---------
     0000  0100   (04)
```

и получается результат: A¤ — HEX (04).

Пример 133.

```
5 DIM A%2
10 A%=HEX(25A4)
20 XOR(A%,F0):HEXPRINT A%

D554
```

При выполнении программы производятся действия:

```
A¤ = 0010  0101  1010  0100   (25A4)
     1111  0000  1111  0000   (F0F0)
     ---------------------
     1101  0100  0101  0100
      D5          54
```

и получается результат: A¤ = HEX (D554).

Оператор BOOL. Структура оператора:

BOOL ⟨шестнадцатеричная цифра⟩ ⟨логические аргументы⟩

Оператор BOOL предназначен для выполнения любой из 16 возможных логических операций. Номер логической операции задается в операторе шестнадцатеричной цифрой от 0 до F.

Результат логических операций (за исключением операции 5) присваивается первой символьной переменной. Результат BOOL 5 присваивается второй символьной переменной.

Коды логических операций приведены в табл. 12.

12. Коды логических операций

Коды	Значение		Результат						
	X	Y							
Двоичные	1 1 0 0	1 0 1 0	0 0 0 0	0 0 0 1	0 0 1 0	0 0 1 1	0 1 0 0	0 1 0 1	0 1 1 0
Шестнадцатеричные коды операций			0	1	2	3	4	5	6

Коды	Результат								
Двоичные	0 1 1 1	1 0 0 0	1 0 0 1	1 0 1 0	1 0 1 1	1 1 0 0	1 1 0 1	1 1 1 0	1 1 1 1
Шестнадцатеричные коды операций	7	8	9	A	B	C	D	E	F

Эти операции принято называть:

0 — нулевая, т. е. результат тождественно равен нулю независимо от значений X и Y;

1 — отрицание логического ИЛИ (НЕ ИЛИ);

2 — логическое И со значениями НЕ X, Y;

3 — НЕ X;

4 — отрицание импликации;

5 — отрицание Y (НЕ Y);

6 — отрицание равнозначности или сложение по модулю 2;

7 — отрицание логического И;

8 — логическое И;

9 — равнозначность или отрицание сложения по модулю 2;

A — тождественное Y; т. е. результат равен значению Y независимо от значения X;

B — импликация;

C — тождественное X; т. е. результат равен значению X независимо от значения Y;

D — логическое ИЛИ значений X и НЕ Y;

E — логическое ИЛИ;

F — тождественное значение 1.

Коды операций, перечисленные в табл. 12, получены применением правил логических операций к двум величинам X и Y (1100 и 1010). Правило получения результата следует рассматривать для каждой пары битов.

Пример 134.

```
5 DIM A$*2
10 A$=HEX$(5432)
20 BOOL 3(A$,00)
30 HEXPRINT A$

ABCD
```

Оператором BOOL задается выполнение логической операции 3 (первая шестнадцатеричная цифра в операторе BOOL) над символьными переменными. При проведении этой операции над двумя величинами X и Y (см. табл. 12) результат равен 0011;

```
X  1100
Y  1010
   ----
   0011
```

Этот результат получен в соответствии со следующими правилами операции:
если бит первого числа равен 1, то результирующий бит равен 0 независимо от значения второго бита;
если бит первого числа равен 0, то результирующий бит равен 1 независимо от значения второго бита.
При выполнении программы производятся действия согласно этим правилам

```
A¤=0101  0100  0011  0010   (5432)
00=0000  0000  0000  0000   (0000)
-------------------------
1010  1011  1100  1101
 AB          CD
```

и получается результат: A¤ = HEX (ABCD).

Пример 135.

```
5 DIM A$*2,B$*2
10 A$=HEX$(4145):B$=HEX$(2185)
20 BOOL 7(A$,B$)
30 HEXPRINT A$

FEFA
```

По коду 7 (в операторе BOOL) проводится логическая операция над переменными:

```
X  1100
Y  1010
   ----
   0111
```

Результат соответствует следующим правилам операции:
если оба бита единичные, то результирующий бит равен 0, в противном случае результат равен 1.

При выполнении программы производятся действия согласно этим правилам:

```
A¤=0100  0001  0100  0101   (4145)
B¤=0010  0001  1000  0101   (2185)
-------------------------
  1111  1110  1111  1010
    FE          FA
```

и получается результат: A¤ = HEX (FEFA).

Оператор ROTATE. Структура оператора:

ROTATE (⟨символьная переменная⟩, ⟨число сдвигов⟩)

⟨число сдвигов⟩: : = 1÷7

Оператор ROTATE предназначен для циклического сдвига влево битов каждого символа (т. е. каждого байта) заданной символьной переменной на указанное число позиций, причем старшие биты становятся на место младших. Перестановка производится во всех символах переменной, включая и конечные пробелы.

Пример 136.

```
5 DIM A¤3
10 A¤=HEX(0123FE)
20 ROTATE (A¤,4)
30 HEXPRINT A¤

1032EF
```

При выполнении программы производится четырехкратный сдвиг битов символьной переменной A ¤:

```
0000  0001  (01)  }
0010  0011  (23)  } исходное состояние
1111  1110  (FE)  }

0001  0000  (10)  }
0011  0010  (32)  } конечное состояние
1110  1111  (EF)  }
```

и получается результат: A ¤ = HEX (1032 EF).

4. ПРЕОБРАЗОВАНИЕ ЧИСЛА В СИМВОЛЬНЫЙ ФОРМАТ И НАОБОРОТ. ОПЕРАТОР CONVERT

Оператор CONVERT имеет две формы:

CONVERT ⟨символьная переменная⟩ **TO** ⟨числовая переменная⟩

CONVERT ⟨а. в.⟩ **TO** ⟨символьная переменная⟩, (⟨формат⟩)

<Формат> — это числовой формат, как в операторе PRINTUSING.

В зависимости от формы оператор CONVERT предназначен для преобразования символьной переменной в числовую и наоборот.

Первая форма оператора CONVERT. Оператор CONVERT преобразует значение символьной переменной в числовое значение и затем присваивает числовой переменной полученное значение.

Значение символьной переменной должно представлять правильное по формату число (целое или действительное). Часть символьной переменной можно преобразовать в числовую, используя функцию STR.

Пример 137.

```
10 A$="1234":B$="-7.50"
20 CONVERT A$TOX:CONVERT B$TOC
30 PRINT X;C

 1234 -7.5
```

Вторая форма оператора CONVERT. Оператор CONVERT преобразует значение числовой переменной по формату, заданному в операторе, и записывает его в символьную переменную в кодах символов КОИ-8 (ГОСТ 19768-74).

Форматы переменных — это числовые форматы, используемые и в операторе PRINTUSING:

формат 1 — число в естественной форме;

формат 2—число в экспоненциальной форме.

Числовые значения формируются согласно следующим правилам:

если формат начинается со знака (+), то в символьной переменной всегда указывается знак (+ или —) в соответствии со знаком числовой переменной;

если формат начинается со знака (—) , то знак (+) в символьной переменной заменяется пробелом, знак (—) указывается;

если в формате нет знака (+ или —), то в символьной переменной знак числа также отсутствует;

если задан формат 1, то значение числовой переменной преобразуется к числу в естественной форме, при этом в соответствии с форматом отбрасываются избыточные цифры или добавляются недостающие нули перед целой частью числа, десятичная точка устанавливается в определенное форматом место; если число превышает отведенный ему формат, то выдается сообщение об ошибке (ERR.36);

если задан формат 2, то значение числовой переменной преобразуется к числу в экспоненциальной форме, символы формата заменяются на E ± XX, где XX — показатель степени при основании 10; нули, стоящие перед целой частью, в формат не включаются.

Вторая форма оператора CONVERT позволяет включать числа в символьную переменную в таком виде, в каком это число необходимо отпечатать; первая форма оператора CONVERT — использовать это число в вычислениях.

Пример 138.

```
10 DIM AX27:INIT (2C)AX:PRINT AX
20 X=42.169:Y=-71.12:CONVERT YTOBX,(-###.#↑↑↑↑)
30 CONVERT XTOSTR(AX,1,4),(## #)
40 CONVERT XTOSTR(AX,6,4),(+###)
50 CONVERT XTOSTR(AX,11,8),(-##.####)
60 CONVERT XTOSTR(AX,20,8),(-#.#↑↑↑↑)
70 PRINT AX:PRINT BX

,,,,,,,,,,,,,,,,,,,,,,,,,
42.1,+042, 42.1690, 4.2E+01
-711.2E-01
```

5. ПРЕОБРАЗОВАНИЕ ЧИСЕЛ ПО ФОРМАТУ И УПАКОВКА В СИМВОЛЬНУЮ ПЕРЕМЕННУЮ ИЛИ МАССИВ. РАСПАКОВКА ЧИСЕЛ

Если в обработке участвуют большие объемы числовых данных, имеющих небольшую разрядность, то для хранения чисел можно использовать символьные переменные. Символьные переменные в этом случае выполняют роль буфера и позволяют сократить объем используемой памяти. Для записи чисел в буфер используется оператор PACK (оператор упаковки). Он преобразует число из действительного или целого формата в формат, указанный в операторе PACK. Чтобы эти числа использовать в вычислениях, их необходимо перевести либо в действительный, либо в целый формат и присвоить соответствующим переменным. Для этой цели используется оператор UNPACK (оператор распаковки) с тем же форматом, который использовался для упаковки.

Оператор PACK. Структура оператора:

PACK (⟨формат⟩) { ⟨символьная переменная⟩ / ⟨метка символьного массива⟩ } FROM ⟨список PACK⟩

⟨**список PACK**⟩ : : = ⟨элемент PACK⟩ [{,⟨элемент PACK⟩}...]

⟨**элемент PACK**⟩ : : = ⟨числовая переменная⟩ | ⟨а. в.⟩

Оператор PACK предназначен для упаковки значений числовых переменных и массивов в символьные по заданному формату.

Число знаков #, используемых в формате, не должно превышать 23.

Символьный массив заполняется, начиная с первого элемента. Если длина символьной переменной или массива мала для хранения всех численных значений, то выдается сообщение об ошибках (ERR 56). При определении необходимой длины символьной переменной или мас-

сива для упаковки данных следует иметь в виду, что каждая числовая переменная упаковывается в соответствии с правилами:

знак # формата соответствует одной цифре и требует 1/2 байт;

если в формате определен знак «+» или «—»,то он занимает вместе со знаком показателя степени старшую тетраду первого байта (1/2 байт);

если формат не имеет знака, то запоминается абсолютная величина числа, а знак показателя степени считается положительным;

положение десятичной точки не сохраняется в памяти; при распаковке данных положение запятой задается форматом распаковки, который должен быть таким же, как и формат упаковки;

упакованная величина всегда занимает целое число байт;

в случае формата 1 (число в естественной форме) значение цифровой переменной упаковывается с отбрасыванием цифр или добавлением нулей в дробной части и, если требуется, добавлением нулей перед целой частью числа согласно заданному формату;

в случае формата 2 (число в экспоненциальной форме) значение цифровой переменной упаковывается без добавления нулей перед целой частью; показатель степени числа занимает один байт.

Приведем примеры записи оператора:

1) 10 PACK (##. ###) A ¤ FROM X

При выполнении этой строки цифровая переменная X упаковывается в символьную переменную A¤ по формату 1 и при этом занимает первые ее три байта;

2) 235 PACK (+ #.##^^^^) A ¤ ( ) FROM N ( ).

При выполнении этой строки программы цифровой массив N ( ) упаковывается в символьный массив A¤ ( ) по формату 2, начиная с первого элемента массива *A*¤ (1). При этом каждое число из массива N ( ) занимает три байта памяти.

Оператор UNPACK. Структура оператора:

UNPACK (⟨формат⟩) { ⟨символьная переменная⟩ / ⟨метка символьного массива⟩ } TO ⟨список PACK⟩

Оператор UNPACK предназначен для распаковки данных, упакованных с помощью оператора PACK, из символьной переменной или массива в цифровые. Для распаковки данных необходимо использовать тот же формат, что и при упаковке. Если упакованных данных не хватает для распаковки, то выдается сообщение об ошибке (ERR 56). Все данные запоминаются последовательно в заданных цифровых переменных и массивах. Массивы заполняются, начиная с первого элемента массива.

Приведем примеры записи оператора:

1) 100 UNPACK (###.##) A¤ TO X, Y, Z

При выполнении этой строки программы данные, упакованные в символьную переменную A ¤ по формату 1, распаковываются и запоминаются в цифровых переменных X, Y, Z.

2) 200 UNPACK (+ #.##∧∧∧∧) A¤( ) TO B( ), Y.

При выполнении этой строки программы данные, упакованные в символьный массив A¤( ) по формату 2, распаковываются и запоминаются в цифровом массиве B( ), кроме последнего числа, которое запоминается в цифровой переменной Y.

6. ПРЕОБРАЗОВАНИЕ ДАННЫХ ИЗ ФОРМАТОВ, ИСПОЛЬЗУЕМЫХ В МАШИНЕ, В ФОРМАТЫ ДЛЯ НЕПОСРЕДСТВЕННОЙ ПЕРЕДАЧИ, И НАОБОРОТ

В преобразованиях используются значения переменных и массивов всех типов данных, а также часть символьного массива, называемая *полем массива*.

Минимальное поле массива — один (любой) символ массива, максимальное поле массива — весь массив, но не менее 8000 символов. В операторах преобразования поле массива указывается в угловых скобках следующими параметрами: номер первого символа поля массива и число символов. Если один из параметров не задан, граница поля совпадает с границей массива. Нумерация символов в массиве сквозная.

В операторах преобразования используются следующие синтаксические обозначения:

⟨**символьный аргумент**⟩: : =⟨символьная переменная⟩|⟨метка символьного массива⟩ [е ⟨⟩полмассива ⟩⟩]

⟨**поле массива**⟩: : = { ⟨номер символа⟩ [,⟨количество символов⟩] | ,⟨количество символов⟩ }

⟨**номер символа**⟩: : = { ⟨арифметическое выражение⟩ | ⟨символьная переменная⟩ }

⟨**количество символов**⟩: : = { ⟨арифметическое выражение⟩ | ⟨символьная переменная⟩ }

⟨**список аргументов**⟩: : = { ⟨переменная⟩ | ⟨метка массива⟩ } [{ , { ⟨переменная⟩ | ⟨метка массива⟩ } } ...]

На параметры, определяющие поле массива, накладываются следующие ограничения:

<номер символа> ⩽K, где K < 8000;

<количество символов> ⩽ K + 1—<номер символа>, где K — число байтов (символов) в массиве.

Если параметры, определяющие поле массива, заданы арифметическими выражениями, то используются их целые части.

Если значение арифметического выражения меньше 1, то выдается сообщение об ошибке (ERR 18).

Если параметры, определяющие поле массива, заданы символьной переменной, то от ее значения используются первые два байта, которые рассматриваются как двоичное число. Десятичный эквивалент этого двоичного числа и есть значение параметра.

Длина символьной переменной, используемой в параметре поле массива, должна быть не менее 2 байт.

Оператор ¤ TRAN. Структура оператора:

¤ **TRAN** (<символьный аргумент 1>, (символьный аргумент 2))
[(маска)] [R]

Здесь <символьный аргумент 1> — преобразуемый массив данных; <символьный аргумент 2> — таблица или список преобразования; <маска> : := <шестнадцатеричная цифра> <шестнадцатеричная цифра>; R — параметр, задающий тип преобразования.

Оператор ¤ **TRAN** предназначен для преобразования кодов. Возможны два типа преобразования.

1. *Параметр* R *указан*. Символьный аргумент 2 используется как список преобразования, в котором попарно записаны байт нового значения и байт старого значения и т. д., последние два байта содержат пробелы (HEX (2020)). Код каждого байта из символьного аргумента замещается кодом байта из символьного аргумента 2 в соответствии со следующим правилом:

в символьном аргументе 2 отыскивается ближайший слева байт, код которого равен коду рассматриваемого байта из символьного аргумента 1;

код байта, предшествующего найденному, из символьного аргумента 2 заносится на место рассматриваемого кода в символьном аргументе 1.

2. *Параметр* R *не указан*. Символьный аргумент 2 используется как таблица преобразования, т. е. код каждого байта из символьного аргумента 1 замещается кодом байта из символьного аргумента 2, номер которого на единицу больше значения кода байта из символьного аргумента 1.

Байты из символьного аргумента 1 просматриваются последовательно слева направо (массивы просматриваются строка за строкой).

Если в операторе ¤ TRAN указан параметр <маска>, то коды байтов из символьного аргумента 1 маскируются, а затем осуществляется поиск в символьном аргументе 2. Маскирование — это логическое умножение кода байта и кода маски (соответствует оператору AND) (см. с. 159).

Если при выполнении оператора в символьном аргументе 2 нет кода байта замены, то код байта (маскированного байта) из символьного аргумента 1 возвращается на свое место.

Чаще всего оператор ¤ TRAN используется для перекодировки разделителей и букв русского алфавита для последующей сортировки по алфавиту.

Пример 139.

```
10 ¤TRAN( A1¤()<7,79>,B¤)
20 ¤TRAN( M¤(),K¤)
30 ¤TRAN( K¤,A¤)R
```

Пример 140.

```
10 DIM A$*10,B$*6
20 A$="A*B,C,E*P*"
30 B$="/*.,  "
40 XTRAN( A$,B$)R
50 PRINT "НОВОЕ ЗНАЧЕНИЕ A$ РАВНО ";A$

НОВОЕ ЗНАЧЕНИЕ A$ РАВНО A/B.C.E/P/
```

Пример 141.

```
10 DIM A$*5,B$*16
20 A$="ИТОГО"
30 B$="0123456789ABCDEF"
40 XTRAN( A$,B$)0F
50 PRINT A$

94F7F
```

Оператор ¤ PACK. Структура оператора:

¤ **PACK** $\left[\left(\left\{\begin{matrix}D\\F\end{matrix}\right\}=\left\{\begin{matrix}\langle\text{символьная переменная}\rangle\\\langle\text{метка символьного массива}\rangle\end{matrix}\right\}\right)\right]$ ⟨символьный аргумент⟩

[, ⟨цифровая переменная⟩] FROM ⟨список аргументов⟩

Оператор ¤ PACK предназначен для последовательной упаковки значений переменных из списка аргументов в символьный аргумент, являющийся буфером для размещения данных, разделителей и другой служебной информации, определяемой форматом упаковки. Формат упаковки значений переменных задается параметрами D, F или умалчивается (не задается в операторе).

При упаковке данных символьный аргумент-буфер должен иметь длину, достаточную для размещения значений всех переменных, указанных в списке аргументов, а также всех необходимых разделителей. В противном случае выдается сообщение об ошибке. При этом значения переменных, находящихся в списке аргументов до значения, упаковка которого привела к ошибочной ситуации, оказываются записанными в символьный аргумент-буфер. Если длина буфера больше, чем требуется для упаковки значений переменных из списка аргументов и разделителей, то неиспользуемые байты сохраняют старые значения.

Реализация оператора ¤ PACK заканчивается, когда исчерпан список аргументов, при этом в цифровую переменную заносится число байтов буфера, использованных при упаковке.

При реализации оператора ¤ PACK массивы упаковываются поэлементно строка за строкой.

Если задан параметр D, то значения переменных упаковываются в разделительный формат:

Зн	Р	Зн	Р	...	Зн	Р

где Зн — значение переменной или элемента массива; Р — (разделитель) — HEX - код второго байта символьной переменной (символьного массива), указанной после «D=».

Разделитель занимает всегда один байт. Длина переменной (массива), указанной после «D=», должна быть не менее двух байтов, при этом массив рассматривается без учета деления на элементы. Первый байт символьной переменной (символьного массива) должен содержать коды HEX (00), HEX (01), HEX (02) или HEX (03), в противном случае выдается сообщение об ошибке. Эта информация не используется оператором ¤ PACK, но необходима для оператора ¤ UNPACK с параметром D, т. е. при распаковке данных, представленных в разделительном формате.

При упаковке в разделительный формат значения переменных представляются:

а) для символьных переменных в виде

С	С	...	С

где С — HEX-код символа. Число байтов, отводимое в буфере под значение символьной переменной (элемента символьного массива), определяется ее длиной, которая может быть определена операторами DIM, COM или может составлять 16 (по умолчанию);

б) для числовых переменных в двух формах: естественной и экспоненциальной в зависимости от значения переменной.

Если значение переменной X находится в диапазоне $0{,}1 \leqslant |X| \leqslant 10^{13}$, то оно представляется в естественной форме и занимает 2—15 байт, так как начальные нули целой части и конечные нули дробной части при упаковке отбрасываются.

Значение числовой переменной в естественной форме имеет вид:

З	Ц	...	Ц	.	Ц	...	Ц

где З — знак числа, если число положительно, то записывается пробел (HEX (20)), если отрицательно, то минус (HEX (2D)); Ц — цифра числа 0—9 (HEX(30) — HEX (39)); . — десятичная точка (HEX(2E)).

Если значение переменной $|X| < 0{,}1$ или $|X| \geqslant 10^{13}$, то оно представляется в экспоненциальной форме, занимает 19 байт и имеет вид:

З	Ц	.	Ц	Ц	Ц	Ц	Ц	Ц	Ц	Ц	Ц	Ц	Ц	Ц	Е	З1	Ц	Ц

где З, Ц — параметры, совпадающие с описанными выше; З1 — знак порядка, если порядок положителен, то плюс (HEX (2B)), если отрицателен — минус (HEX (2D)); Е — признак порядка числа (HEX (45)).

Пример 142. Запись правильного синтаксиса:

```
10 xPACK (D=Ax)Kx FROMA,Bx,X,Cx()
20 xPACK (D=Kx()Mx()FROMA,B(),Cx,K
30 xPACK (D=Kx()<7,114>)Zx()FROMA,B()
```

Пример 143.

```
10 DIM Bx26,Ax4,Mx2
20 Ax="ABC":X=-42:K=88.24E-21
30 Mx=HEX(002C)
40 xPACK (D=Mx)Bx FROMX,Ax,K:PRINT Bx

-42,ABC , 8.82400000E-20,
```

Заданная длина Bx позволила записать лишь 9 разрядов дробной части числа. Незначащие нули отбросились при записи значений.

В качестве разделителя используется запятая (HEX (2C)).

Если задан параметр F, то значения переменных упаковываются в полевой формат, т. е. каждому значению переменной или элемента массива из списка аргументов отводится свое поле в символьном аргументе-буфере. Данные в буфере представляются в виде:

Поле значения 1	Поле значения 2	...	Поле значения К

Тип и длина каждого значения из списка аргументов задается соответственно двумя байтами символьной переменной (символьного массива), указанной после «F=». При этом К-й переменной (метке массива) из списка аргументов соответствуют байты символьной переменной (массива) с порядковыми номерами (2К —1) и 2К, если нет пропуска поля; символьный массив в этом случае рассматривается без учета деления на элементы.

Итак, пара байтов характеризует поле значения для переменной из списка аргументов. Первый байт задает тип поля значения и может быть равным HEX-коду согласно данным табл. 13.

Второй байт задает длину поля значения. Число байтов, отводимое под поле значения, определяется десятичным эквивалентом HEX-кода, содержащегося в этом байте.

Каждая переменная (метка массива) из списка аргументов должна иметь свое описание (два байта) в символьной переменной (массиве); при этом описание для метки массива определяет поле значения для каждого элемента массива, а не для всего массива в целом.

13. Значения байта — тип поля значения

HEX-код байта — тип поля значения	Соответствие формату
00	Пропуск в символьном аргументе количества байтов, указанного во втором байте (в описании длины поля значения)
10	Числовой свободный формат (совпадает с D); цифре соответствует 1 байт
2X	Числовой десятичный формат (естественная форма); цифре соответствует 1 байт
5X	Числовой упакованный формат (естественная форма); цифре соответствует одна тетрада
A0	Формат для символьных данных

Примечание. X — шестнадцатеричная цифра, десятичный эквивалент которой определяет положение запятой в числе от правого края.

Пропускаемое при записи поле также описывается двумя байтами. Значение первого байта равно HEX (00). При выполнении оператора байты пропускаемого поля сохраняют прежнее значение.

При упаковке данных по оператору ¤ PACK с параметром F тип переменных (меток массивов) и тип поля значения должны совпадать, т. е. символьные переменные (массивы) должны упаковываться в формат символьных данных, а числовые — в форматы числовых данных (см. табл. 13). В противном случае выдается сообщение об ошибке.

Если поле значения имеет недостаточную длину, то символьное данное усекается справа только в дробной части. Если поле значения недостаточно для размещения целой части числа, то выдается сообщение об ошибке.

Если поле значения больше, чем необходимо для размещения значения, то свободные байты поля значения заполняются пробелами и располагаются после значения в случае форматов A0 и 10, а в случае форматов 2X и 5X заполняются нулями, которые могут располагаться как до, так и после значения согласно указанию о местоположении запятой в числе.

Оператор ¤ PACK с параметром F упаковывает данные в символьный и числовые форматы в соответствии с заданным в операторе типом поля значения (см. табл. 13).

Символьный формат (тип поля значения A0)

C	C	...	C

Здесь C — HEX-код символа.

Числовой свободный формат (тип поля значения 10) совпадает с числовыми формами данных для разделительного формата оператора ¤ PACK с параметром D.

Числовой десятичный формат (тип поля значения 2X)

З	Ц	Ц	...	Ц

Здесь З — знак числа; если число положительно, то плюс (HEX (2B)), отрицательно — минус (HEX (2D)); Ц — цифра числа 0—9 (HEX (30)—HEX(39). Число в естественной форме.

Числовой упакованный формат (тип поля значения 5X)

цц	...	цц	цз

Здесь ц — цифра числа 0—9, представленная в символьном аргументе двоичной тетрадой; з — знак числа; если число положительно, то З = C_{16}, отрицательно — З = D_{16}.

Местоположение десятичной запятой в числе от правого края поля в форматах 2X и 5X определяется второй тетрадой байта — типом поля значения.

Пример 144. Запись правильного синтаксиса:

```
10 xPACK (F=Fx)RxFROMX,Y,Ax()
20 xPACK (F=Fx())Hx()FROMX,Y,A(1,2)
```

т. е. так же, как и для разделительного формата ¤ PACK.

Пример 145.

```
10 DIM Bx60
20 Ax="ABC":X=-12:Y=1.2345:Z=12.345:INIT (FF)Bx
30 Fx=HEX(A005200510052400002530552065506)
40 xPACK (F=Fx)BxFROMAx,X,Y,Z,Z,Z,Z
50 HEXPRINT Bx

41424320202D3030313220312E32332B3030303132343530FFFF000012
345C00000001234C00001234500CFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
```

После выполнения программы будет индицироваться следующий результат:

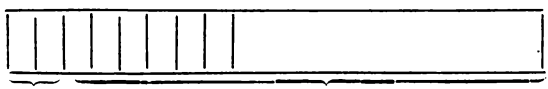
41424320202D3030313220312E32332B3030303132343530FFF000012345C0000000 1234C00001234500CFFFFFFFFFFFF.

Если параметры D и F отсутствуют, то данные по оператору ¤ PACK упаковываются в стандартный формат представления данных на внешних носителях (лентах, дисках) по операторам ДАТА SAVE DA, ДАТА SAVE, DATA SAVE DC, т.е. в виде:

К	Разделитель данных	Данное	Разделитель данных	...	Разделитель данных	Данное

Здесь K — контрольный байт (этот байт не используется операторами ¤ PACK и ¤ UNPACK, а лишь дает системе информацию о начале блока данных).

Двухбайтный разделитель данных содержит информацию о типе следующих за ним данных и их длине в байтах. При этом информация в разделителе формируется следующим образом:



Тип значения Длина значения

Здесь старшие 2 бит первого байта задают тип значения (00— цифровое значение, 01 —символьное значение), остальные 6 бит этого байта и следующий за ним байт содержат длину значения в байтах.

При выделении места под символьный аргумент необходимо учитывать место, отводимое под контрольные байты и разделители данных.

Оператор ¤ PACK без параметров упаковывает данные в соответствии с форматами представления их на внешних носителях.

Приведем примеры записи оператора:

```
10 ¤PACK B¤FROM¤,B,P¤,A¤()
20 ¤PACK B¤()<10,100>FROMB1¤<A,5),X()
```

Оператор ¤ UNPACK. Структура оператора:

¤ **UNPACK** [(({D / F} = {⟨символьная переменная⟩ / ⟨метка символьного массива⟩})) ⟨символьный аргумент⟩,
[, ⟨числовая переменная⟩] TO ⟨список аргументов⟩

Оператор ¤ UNPACK предназначен для распаковки данных из символьного аргумента-буфера и последовательного присвоения распакованных значений переменным (массивам) из списка аргументов.

Формат, в котором упакованы данные в символьном аргументе, должен совпадать с форматом, тип которого задается параметрами F и D или их отсутствием в операторе ¤ UNPACK. При несовпадении либо будет выдано сообщение об ошибке, либо данные после распаковки будут искажены. Обычно во избежание искажения данных оператор ¤ UNPACK используется для распаковки символьного аргумента, данные в который были помещены по оператору ¤ PACK, при этом параметры операторов, определяющие тип формата, должны совпадать.

Реализация оператора ¤ UNPACK заканчивается, если исчерпан весь список аргументов или если исчерпан буфер (подробнее см. описание каждой из форм оператора ¤ UNPACK).

Если в списке аргументов стоит метка массива, то при реализации оператора ¤ UNPACK данные, распакованные из символьного аргумента, заносятся в массив поэлементно строка за строкой.

Если при распаковке символьного аргумента возникает ошибка, то часть переменных (массивов) из списка аргументов, находящихся до переменной, при заполнении которой возникла ошибка, имеют новые (распакованные) значения, а остальные сохраняют старые значения.

В числовую переменную по окончании выполнения оператора ¤ UNPACK заносится число использованных в буфере байтов.

Если задан параметр D, то при распаковке значений предполагается, что они упакованы в символьном аргументе в разделительном формате, т. е. занесены в него по оператору ¤ PACK с параметром D, или получены другим способом, но представлены в форматах, указанных в операторе ¤ PACK с параметром D (см. с. 171).

Первый байт символьной переменной (массива), указанной после «D =» может принимать значение в соответствии с данными табл. 14. Значением второго байта является HEX-код разделителя данных, при этом значение символьной переменной (массива) должно совпадать со значением символьной переменной (массива), указанной ранее при упаковке данных в операторе ¤ PACK с параметром D.

Переменные, стоящие в списке аргументов, но не используемые (пропускаемые), сохраняют старые значения.

При распаковке символьного значения по оператору ¤ UNPACK длина распакованного значения равна меньшей из двух длин — длины переменной, в которую заносится распакованное значение, и длины упакованного значения. Если длина упакованного значения меньше длины переменной, то распакованные значения дополняются пробелами; если больше, то распаковываемое значение усекается справа.

Конец значения определяется по первому встретившемуся разделителю, поэтому код разделителя не должен совпадать с кодами символов значения.

14. Значения HEX-кода первого байта символьной переменной

Значение HEX-кода первого байта	Правила распаковки по оператору ¤ UNPACK (разделительный формат)
00	Ошибка, если данных в символьном аргументе недостаточно для всего списка аргументов. Дополнительный разделитель в символьном аргументе означает пропуск из списка аргументов (дополнительный разделитель есть, если между разделителями нет данных)
01	Если данных в символьном аргументе недостаточно для всего списка аргументов, то оставшиеся элементы из списка аргументов игнорируются. Дополнительный разделитель в символьном аргументе означает пропуск элемента из списка аргументов
02	Ошибка, если данных в символьном аргументе недостаточно для всего списка аргументов. Дополнительные разделители в символьном аргументе игнорируются
03	Если данных в символьном аргументе недостаточно для всего списка аргументов, то оставшиеся элементы из списка аргументов игнорируются. Дополнительные разделители в символьном аргументе игнорируются

При распаковке числового значения дополнительные пробелы, встречающиеся в числовом значении, игнорируются.

Числовые значения всегда могут быть распакованы в символьную переменную, а символьные значения в числовую лишь при соблюдении всех ограничений на формат числового значения (в противном случае выдается сообщение об ошибке).

Приведем пример записи правильного синтаксиса:

```
10 ×UNPACK (D=B×)K×TOX,A(),B×
20 ×UNPACK (D=P×())K×TOX()
```

Пример 146.

```
10 DIM B×26,A×4,M×2
20 B×=" 42,ABC , 8.824000000E-20,"
30 M×=HEX(002C)
40 ×UNPACK (D=M×)B×TOX,A×,K
50 PRINT B×:PRINT X,A×,K

 42,ABC , 8.824000000E-20,
 42              ABC              8.824000000E-20
```

Пример 147.

```
10 DIM F×12,K×2
20 K×=HEX(032C)
30 F×="ABC,DEF,,GHI"
40 ×UNPACK (D=K×)F×TOM×,A×,B×,C×
50 PRINT M×,A×,B×,C×

ABC             DEF             GHI
```

Если задан параметр F, то при распаковке значений предполагается, что они упакованы в символьном аргументе в полевом формате, т. е. занесены в него по оператору ¤ PACK с параметром F или получены другим способом, но имеют формат представления, аналогичный полученному по оператору ¤ PACK с параметром F.

Распаковка значений из символьного аргумента в список аргументов происходит в соответствии с информацией, хранящейся в символьной переменной (массиве), где байт с порядковым номером (2K—1) задает тип поля упаковки, а байт с порядковым номером 2K задает длину, отведенную в символьном аргументе под значение K-й переменной в списке аргументов (подробнее см. оператор ¤ PACK, полевая форма). При несоответствии форматов и длин значений, задаваемых соответствующими байтами в символьной переменной (массиве), может

произойти искажение информации, заносимой в переменную из списка аргументов, или может быть выдано сообщение об ошибке. Если при распаковке в символьную переменную длина значения, выбираемая из символьного аргумента, больше длины переменной, которой это значение присваивается, то значение усекается; если длина значения меньше, то оно дополняется пробелами.

Для обеспечения возможности распаковки данных форматы представления данных должны соответствовать приведенным в операторе ¤ PACK, причем могут быть следующие отклонения:

во всех числовых форматах допустимы дополнительные пробелы, которые игнорируются при распаковке по оператору ¤ UNPACK;

в формате 5X код знака плюс может содержать коды A и E вместо кода C, а код минуса код B вместо D.

Приведем пример записи правильного синтаксиса:

```
10 ¤UNPACK (F=F¤)R¤TOM¤,X¤,Y¤,Z¤
20 ¤UNPACK (F=A¤())P¤()<10,20>TOM,X,Y¤,Z()
30 ¤UNPACK (F=F¤)P¤,KTOM¤,X¤,Y¤,Z¤
```

Пример 148.

```
10 DIM R¤37,A¤9,C¤3
20 R¤="I.KMIIH M 5.80130 1234.56789"
30 F¤=HEX(A008A001100710031010)
40 ¤UNPACK (F=F¤)R¤TOA¤,C¤,X,Y,Z
50 PRINT A¤,C¤,X,Y,Z

'.KMIIH          M                  5.8013   1        234.56789
```

Оператор ¤ UNPACK допускает отклонения от форматов. Хотя упакованные числа без знака, распаковка все-таки произведена.

Пример 149.

```
10 DIM R¤21
20 R¤="+1234567"
30 F¤=HEX(2408A005A005)
40 STR(R¤,9,10)=HEX(41424344454647484950)
50 ¤UNPACK (F=F¤)R¤TOX,Y¤,Z¤
60 PRINT X,Y¤,Z¤

 123.4567        ABCDE            FGHIP
```

После выполнения программы на экране будут получены результаты:

123. 4567 ABCDE FGHIP

Если в операторе ¤ UNPACK параметры F и D отсутствуют, то данные в символьном аргументе должны находиться в стандартном формате представления данных на внешних носителях (см. оператор ¤ PACK без параметров), в противном случае выдается сообщение об ошибке.

При распаковке необходимо соблюдать соответствие между типом данных, упакованных в символьный аргумент, и типом переменных, используемых для распаковки, т. е. символьные данные должны заноситься в символьные переменные (массивы), а числовые — в числовые, в противном случае выдается сообщение об ошибке.

При распаковке символьного значения число символов, заносимых в символьную переменную, определяется наименьшей из длины распаковываемого значения и длины переменной. Если длина значения больше, то оно усекается, если меньше, то дополняется пробелами до длины символьной переменной.

Приведем примеры записи правильного синтаксиса:

```
10 ¤UNPACK M¤TOA,B¤,C()
20 ¤UNPACK A1¤()<20,50>TOB¤()
```

Глава XIV
РАБОТА С МАГНИТНЫМИ ЛЕНТАМИ

1. ОБМЕН С КАССЕТНЫМ НАКОПИТЕЛЕМ НА МАГНИТНОЙ ЛЕНТЕ

Кассетный накопитель на магнитной ленте (КНМЛ) является консольным ленточным устройством с ФАУ, равным 08, и входит в 5-е и 6-е исполнения машины. КНМЛ позволяет записывать (считывать) информацию на двухдорожечную МЛ.

Информация на МЛ имеет файловую структуру, но, в отличие от каталогизированного диска, КНМЛ не имеет УКФ (см. с. 116).

Файлы делятся на программные и на файлы данных. Программный файл формируется одним оператором вывода и содержит одну логическую запись. Файл данных формируется так же, как на каталогизированном диске.

Записывается и считывается информация только при движении ленты вперед (от начала к концу).

Для выполнения ввода-вывода используются специальные ленточные операторы.

2. ОБМЕН С НАКОПИТЕЛЕМ НА МАГНИТНОЙ ЛЕНТЕ

Структура информации на магнитной ленте. Накопитель на магнитной ленте (НМЛ) служит для записи данных на 9-дорожечную магнитную ленту.

Запись и считывание с ленты производятся по зонам переменной длины. Зона содержит от 19 до 1024 девятиразрядных кода, из которых

девятый бит является контрольным. После данных записываются байт циклического контроля и байт продольного контроля. Расстояние между ними и до блока данных в 4 раза больше расстояния между соседними байтами в информационном массиве. Байт продольного контроля образуется путем поразрядного сложения по модулю 2 всех информационных байтов. Байт циклического контроля образуется аналогично, но с циклическим сдвигом информационных байтов. Оба контрольных байта подсчитываются и выводятся автоматически.

Зоны данных разделены межзонными промежутками, длина которых может быть в интервале 12,7—7600 мм.

Для организации иерархической структуры данных (логическая запись, файл), на магнитную ленту в качестве разделителя зон может быть записана зона специальной структуры — ленточный маркер. Ленточный маркер составляет один байт, его значение HEX (13). Например, конец логической записи можно отмечать одним ленточным маркером, конец файла — двумя.

В начале и конце ленты с помощью светоотражающих полосок нанесены маркеры начала и конца ленты (физические маркеры).

Обмен с НМЛ программируется на уровне процедур БИФ «Искра 015-25».

Процедуры БИФ «Искра 015-25». Процедура БИФ — это одно управляющее слово в HEX-кодах, выполняемое после его вывода в БИФ. Программы реализации процедур обмена записаны в памяти микропроцессора БИФ.

БИФ выполняет следующие процедуры:

начальной установки;

установки режима работы НМЛ;

обмена данными (ввода-вывода).

Процедура начальной установки служит для приведения БИФ в исходное состояние и содержит три байта;

1-й байт — заголовок процедуры, равный 1 В;

2-й байт — идентификатор процедуры, равный 13 при работе без проверки контрольной суммы (КС) и 93 при работе с проверкой КС (всегда к идентификатору добавляется код 80, если процедура должна выводиться с КС);

3-й байт — байт контрольной суммы, равный 00, если КС не подсчитывается, и 88, если КС подсчитывается.

КС здесь и в других процедурах есть сумма по модулю 2 всех байтов, входящих в процедуру.

Процедура установки режима работы НМЛ состоит из четырех байтов;

1-й байт — заголовок процедуры, равный 1В;

2-й байт — идентификатор процедуры, равный 00 без подсчета КС и 80 с подсчетом КС;

3-й байт — задание режима работы (01 — чтение, 02 — пропуск одной зоны данных при движении назад, 03 — запись одной зоны дан-

ных, 04 — запись ленточного маркера, 05 — стирание одной зоны данных, 06 — перемотка ленты в начало;

4-й байт — контрольная сумма первых трех байтов процедуры, если был указан признак КС (80) во втором байте, или 00, если признак не указан.

Например,

HEX (1B000300) — процедура задает режим записи информации;
КС процедуры не подсчитывается;
HEX (1B80019A) — процедура задает режим чтения информации. КС подсчитывается.

Процедура ввода-вывода задает длину порции обмена, называемую длиной посылки и состоит из пяти байтов;

1-й байт — заголовок процедуры, равный 1B;

2-й байт — идентификатор процедуры, равный 01 без подсчета КС и 81 с подсчетом КС;

3-й байт — старший байт длины посылки;

4-й байт — младший байт длины посылки;

5-й байт — байт КС или 00.

Следовательно, длина посылки содержит два байта.

Нумерация байтов информации при передаче в БИФ начинается с 0 и длина посылки берется на единицу меньше фактической.

Максимальное число байтов, передаваемое в БИФ, определяется физической емкостью его буфера и составляет 1024 байт.

Длина посылки вычисляется следующим образом: число байтов без 1 переводится в двоичную систему счисления, разбивается на тетрады и записывается в процедуру двумя HEX-кодами в виде значений 3-го и 4-го байтов.

Приведем примеры расчета длины посылки:

1) число передаваемых байтов равно 1024. Вычислим длину посылки L:

$$L = 1024 - 1 = 1023.$$

Получаем двоичное число и разбиваем его на тетрады:

0000 0011 1111 1111

или в HEX-кодах: 03 FF.

Значит, максимальная длина посылки L = 03 FF.

Процедура запишется следующим образом: HEX (1B0103FF00).

2) Число передаваемых байтов равно 19. Вычислим длину посылки:

$$L = 19 - 1 = 18.$$

Полученное число переводим в двоичную систему счисления и разбиваем на тетрады:

0001 0010—12 — это младший байт длины посылки. Старший байт запишется в виде 00.

Процедура: HEX (1B01001200).

Процедура ввода-вывыда не изменяет установленный ранее режим работы НМЛ.

Рассмотрим последовательность процедур для ввода одной физической записи информации:

1) процедура начальной установки, т. е. символьной переменной P ¤ присваивается значение процедуры: P ¤ = HEX (1B1300). Затем оператором DATA SAVE BT P ¤ она передается в БИФ.

Процедура начальной установки выполняется перед первым обращением к НМЛ для обмена. В остальных случаях следует начинать с процедуры установки режима;

2) процедура установки режима

```
P1 ¤ = HEX (1B000300)
DATA SAVE BT P1 ¤
```

3) процедура ввода-вывода:

```
P2 ¤ = HEX (1B01XXXX00)
DATA SAVE BT P2 ¤
```

4) вывод информации оператором DATA SAVE BT.

Аналогично выполняется чтение информации с МЛ, только используется оператор DATA LOAD BT вместо оператора DATA SAVE BT.

Стирание предыдущего блока информации в процессе работы с МЛ выполняется в определенной последовательности:

1) выполняется возврат на один блок (процедура установки режима)

```
P1 ¤ = HEX (1B000200)
DATA SAVE BT P1 ¤
```

2) задается режим стирания (процедура установки режима)

```
P2 ¤ = HEX (1B000500)
DATA SAVE BT P2 ¤
```

3) задается длина стираемого блока (процедура ввода-вывода)

```
P3 ¤ = HEX (1B01XXXX00)
DATA SAVE BT P3 ¤
```

Примечание. XXXX — шестнадцатеричное значение длины посылки

Операторы ввода-вывода данных на МЛ. Оператор DATA LOAD BT имеет следующую структуру:

DATA LOAD BT [⟨устройство⟩,] { ⟨символьная переменная⟩ / ⟨метка символьного массива⟩ }

Оператор предназначен для ввода информации с внешнего устройства в указанную символьную переменную или символьный массив.

Число вводимых байтов определяется длиной заданной символьной переменной (массива).

Оператор DATA SAVE BT имеет следующую структуру:

DATA SAVE BT [R] [⟨устройство⟩,] { ⟨символьная переменная⟩ / ⟨метка символьного массива⟩ }

Оператор предназначен для записи на МЛ данных в виде физической записи.

Параметр R означает перезапись, т. е. перед выполнением новой записи МЛ автоматически возвращается назад на одну запись.

Оба оператора являются универсальными. В зависимости от заданного ФАУ устройства они могут обеспечивать работу с перфораторами, печатающим устройством и другими устройствами, допускающими процедурный обмен.

Запись и считывание программы. Программа может быть записана на магнитную ленту только через область данных оперативной памяти (ОП). Сначала оператором SAVE программа записывается в символьную переменную или массив, откуда оператором DATA SAVE BT переписывается на ленту.

Структура оператора SAVE:

SAVE <символьная переменная> [P] [<номер строки>] [, <номер строки>]

В символьную переменную записывается программа или ее часть, ограниченная заданными номерами строк, с признаком защиты или без него. Таким образом, программа оформляется в виде данных.

Пример 150.

```
10 DIM P ¤ (4) 250
20 SAVE P¤ (1), 1, 3
30 SAVE P¤ (2), 4,7
40 SAVE P¤ (3) 8, 10
50 SAVE P¤ (4) 11
    .
    .
100 DATA SAVE BT P¤ ( )
```

В символьный массив P¤ (0) записались 11 строк программы и вывелись на ленту в виде блока данных.

При чтении оператором DATA LOAD BT программа вводится с ленты в область данных оперативной памяти, затем оператором LOAD загружается из переменной в область программы.

Структура оператора LOAD:

LOAD <символьная переменная> [<номер строки>] [, <номер строки>]

Из символьной переменной программа или ее часть переписывается в зону программы.

Исполняется оператор аналогично оператору LOAD DC.

Пример 151. Обращение к НМЛ для записи и чтения информации:

```
10 REM  ЗАПИСЬ ИНФОРМАЦИИ НА МЛ
20 DIM D$3,S$(6)4,E$5,A$(3)85,B$(3)85
30 SELECT TAPE1B(7999)
40 D$=HEX(1B1300)
50 S$(1)=HEX(1B000100)
60 S$(2)=HEX(1B000200)
70 S$(3)=HEX(1B000300)
80 S$(4)=HEX(1B000400)
90 S$(5)=HEX(1B000500)
100 S$(6)=HEX(1B000600)
110 E$=HEX(1B0100FF00)
120 A$(1)="ПРИМЕР КОНТРОЛЯ ОБМЕНА С НМЛ"
130 A$(2)="ВЫПОЛНЕН С НМЛ СМ 5300-01"
140 A$(3)="НА ПЭКВМ 'ИСКРА 226'"
150 INIT (2A)B$()
160 DATA SAVE BT D$
170 DATA SAVE BT S$(5)
180 DATA SAVE BT S$(6)
190 DATA SAVE BT S$(3)
200 DATA SAVE BT E$
210 DATA SAVE BT B$()
220 STOP "ЗАПИСЬ ЗАКОНЧЕНА, ДЛЯ ПРОДОЛЖЕНИЯ СЧЕТА НАЖМИТЕ КЛ
ВИШУ 'CONTINUE'"
230 REM  СЧИТЫВАНИЕ ИНФОРМАЦИИ С МЛ
240 DATA SAVE BT S$(2)
250 DATA SAVE BT S$(1)
260 DATA SAVE BT E$
270 DATA LOAD BT B$()
280 DATA SAVE BT S$(6)
290 PRINT /0C,B$()
```

Глава XV

СОРТИРОВКА

1. ОБЩИЕ СВЕДЕНИЯ

Для сортировки данных в языке БЕЙСИК имеются специальные операторы. Они реализуют сортировку двух типов:

полную сортировку массива с неупорядоченными (произвольными) значениями по возрастанию значений;

поэтапную сортировку массива, в строках которого значения элементов упорядочены (рассортированы) по возрастанию, т. е. слияние рассортированных строк.

Сортировка символьных данных по возрастанию означает сортировку по возрастанию кодов символов. Например, код каждой буквы русского алфавита больше кода буквы латинского алфавита; код латинской буквы A меньше кода латинской буквы B; коды цифр меньше кодов букв (см. табл. 7).

Сортировка реализуется операторами MAT CONVERT, MAT SORT, MAT MERGE, MAT MOVE. Однако в языке БЕЙСИК есть еще два оператора, которые по синтаксической структуре можно отнести к той же группе операторов, что и операторы сортировки, хотя использовать их в сортировке не обязательно. Это операторы:

MAT COPY — оператор побайтной переписи символьной информации из одного массива в другой;

MAT SEARCH — оператор, используемый для получения массива-локатора, содержащего совокупность номеров символов массива, удовлетворяющих условию, заданному в операторе.

У всех перечисленных операторов есть имя, начинающееся символами MAT (как у матричных операторов). По синтаксису они отличаются от матричных операторов. Массивы в операторах этой группы обозначаются меткой массива, переопределение размерности не выполняется. Объектом обработки для операторов является символьный массив или его часть, которая может быть представлена в следующем виде:

цепочка подряд расположенных символов в любой части массива — поле массива, задаваемое номером первого символа цепочки (поля массива) и числом символов (длиной поля массива); длина поля массива ограничена длиной всего символьного массива; нумерация символов в массиве сквозная, независящая от деления на элементы; для указания поля массива используются угловые скобки;

совокупность цепочек подряд расположенных символов из каждого элемента массива; такая совокупность задается параметром <поле элемента>; используется нумерация символов в пределах одного элемента; из каждого элемента массива используются символы с одним и тем же номером в соответствии с заданным полем элемента; поле элемента указывается номером первого символа и числом символов; длина поля элемента ограничена длиной элемента символьного массива; для указания поля элемента используются круглые скобки; номер символа и число символов задаются арифметическими выражениями.

Поле массива и поле элемента записываются обязательно после метки символьного массива:

<поле массива> : := <а. в. 1>, <а. в. 2> — в угловых скобках;

<поле элемента>: := <а. в. 3>, <а.в.4>— в круглых скобках.

Если обозначить M и N число строк и число столбцов символьного массива, длину элемента K (они были указаны в DIM при объявлении символьного массива), то можно записать верхнюю границу диапазона принимаемых значений арифметических выражений:

$$<\text{а. в. } 1> \leqslant K \times M \times N, <\text{а. в. } 2> \leqslant K \times M \times N - <\text{а. в. } 1> + 1$$

$$<\text{а. в. } 3> \leqslant K, <\text{а. в. } 4> \leqslant K - <\text{а. в. } 3> + 1$$

При этом нижняя граница диапазона равна единице.

Приведем примеры правильного синтаксиса:

A¤ ( )<7,2*4.5> — задано поле массива (7,2*4.5) B ¤ ( ) (3,6) — задано поле элемента (в каждом элементе массива по шесть символов, начиная с третьего).

В операторах сортировки используются символьные массивы специального назначения, которые можно охарактеризовать следующим:

массив-локатор — одномерный или двумерный символьный массив с длиной элемента 2 байт, требования по размерности зависят от оператора, использующего массив-локатор;

рабочий массив 1 — одномерный символьный массив с длиной элемента 1 байт, число элементов на единицу больше числа строк сортируемого массива; используется в операторе MAT MERGE для хранения информации о сортируемом массиве, т. е. элементы рабочего массива 1 являются счетчиками используемых в сортировке элементов строк сортируемого массива (номер строки равен номеру элемента рабочего массива 1); последний элемент рабочего массива 1 — признаковый;

рабочий массив 2 — одномерный символьный массив с длиной элемента 2 байт; является рабочей зоной операторов MAT SORT, MAT MERGE; массив не содержит полезной информации для пользователя;

сортируемый массив может содержать не более 254 строк и не более 254 столбцов.

2. СОРТИРОВКА ПЕРВОГО ТИПА

Сортировка программируется несколькими последовательно выполняемыми операторами. При записи программы необходимо соблюдать последовательность:

объявить рабочий массив 2, массив-локатор с числом элементов, не меньшим числа элементов сортируемого массива, и все массивы, необходимые для сортировки;

если исходный массив числовой, его необходимо преобразовать оператором MAT CONVERT к виду, удобному для сортировки, т. е. в специальный символьный формат;

получить массив-локатор оператором MAT SORT, после чего можно либо сразу же приступить к упорядочению, либо сохранить массив, а упорядочение выполнить в другой программе;

упорядочить исходный массив оператором MAT MOVE в соответствии с массивом-локатором;

упорядочить остальные массивы, если для них способ упорядочения задается тем же массивом-локатором.

Оператор MAT CONVERT. Структура оператора:

MAT CONVERT<метка цифрового массива>TO<метка символьного массива> [(<поле элемента>)].

Оператор MAT CONVERT предназначен для преобразования элементов числового массива в элементы (поля элементов) символьного

массива в формате, удобном для использования операторов сортировки MAT SORT и MAT MERGE.

Каждый элемент цифрового массива занимает восемь байтов в элементе символьного массива, распределенных следующим образом:

Знак	Показатель степени	Мантисса

Знак занимает 1/2 байт и может иметь следующие значения:
9 — мантисса и показатель положительны;
8 — мантисса положительна, показатель отрицателен;
1 — мантисса и показатель отрицательны;
0 — мантисса отрицательна, показатель положителен.

Показатель степени занимает 1 байт (8 бит) и содержит показатель степени в прямом коде, если мантисса и показатель имеют одинаковые знаки, или в дополнительном коде, если они имеют разные знаки.

Остальные 6,5 байт занимает мантисса в прямом коде, если ее знак положителен, или в дополнительном, если знак отрицателен.

Если длина элемента (поля элемента) символьного массива больше 8 байт, то байты каждого элемента (поля элемента), начиная с девятого, заполняются пробелами, если длина меньше 8 байт, то младшие значащие цифры мантиссы отбрасываются.

Длина элемента (поля элемента) символьного массива должна быть не менее 2 байт, в противном случае происходит искажение информации.

Для реализации оператора необходимо, чтобы число элементов в символьном массиве было не меньше, чем в числовом, в противном случае выдается сообщение об ошибке.

Реализация оператора заканчивается, если исчерпан числовой массив; при этом если число элементов символьного массива больше, чем числового, то элементы символьного массива, в которые не заносятся значения элементов числового массива, сохраняют старые значения.

Пример 152. Правильный синтаксис:

```
10 MAT CONVERT A()TOA$()
20 MAT CONVERT L()TOM$()(3,5*(X-Y))
```

Оператор MAT SORT. Структура оператора:

MAT SORT <метка символьного массива> TO <метка рабочего массива 2>, <метка массива-локатора>

Оператор MAT SORT предназначен для построения массива-локатора, используемого в операторе MAT MOVE и содержащего координаты местоположения элементов сортируемого массива в порядке возрастания значений элементов.

Массив-локатор (одномерный или двумерный символьный массив) должен иметь не меньшее число элементов, чем сортируемый массив (в противном случае выдается сообщение об ошибке); длина каждого элемента этого массива должна быть 2 байт.

Рабочий массив 2 — одномерный символьный массив с числом элементов не менее, чем в сортируемом массиве; каждый элемент, длина которого равна 2 байт, служит рабочей зоной оператора.

Сортируемый массив — любой символьный массив.

Выполнение оператора заканчивается, когда просмотрен (отсортирован) весь исходный массив данных. Если число элементов в массиве-локаторе больше, чем в сортируемом массиве, то в первый неиспользуемый элемент массива-локатора заносится признак конца значений HEX (0000).

Если в сортируемом массиве находятся одинаковые элементы, то порядок следования их координат в массиве-локаторе может не совпадать с расположением элементов в сортируемом массиве.

В результате выполнения оператора первый элемент массива-локатора имеет значение, равное номеру строки (первый байт) и номеру столбца (второй байт) элемента сортируемого массива, который в результирующем массиве будет первым. Этот элемент имеет наименьшее значение. Второй элемент массива-локатора содержит координаты элемента, который будет вторым и т. д.

Координата (номер строки или номер столбца) — двухразрядное шестнадцатеричное число.

Оператор можно записать в виде

```
10 MAT SORT A¤ (  ) TON2¤ (  ), L¤ (  )
```

Пример 153.

```
10 DIM A(1,5),A1(1,5),S$(1,5)8,L$(1,5)2,R2$(5)2,M$(1)3
20 MAT READ A
30 MAT CONVERT A()TOS$()
40 MAT SORT S$()TOR2$() L$()
50 MAT MOVE A(),L$(1,1)TOA1(1,1)
60 PRINT "ИСХОДНЫЙ МАССИВ";TAB(17);
70 MAT PRINT A,M$
80 PRINT "МАССИВ-ЛОКАТОР ";TAB(17);
90 HEXPRINT L$()
100 PRINT "РАССОРТИРОВАН-":PRINT "НЫЙ МАССИВ";TAB(17);
110 MAT PRINT A1;M$
120 DATA 14,12,11,15,13


ИСХОДНЫЙ МАССИВ   14  12  11  15  13


МАССИВ-ЛОКАТОР   0103010201050101 0104
РАССОРТИРОВАН-
НЫЙ МАССИВ        11  12  13  14  15
```

Итак, непосредственная сортировка выполняется в строках 30, 40, 50.

Оператор MAT MOVE. Оператор имеет две основные формы:

1) для числовых массивов:

MAT MOVE <метка числового массива>, <элемент массива-локатора> [;<числовая переменная>] TO <элемент числового массива>

2) для символьных массивов:

MAT MOVE <метка символьного массива> [(< поле элемента>)], <элемент массива-локатора> [, <числовая переменная>] TO <элемент символьного массива>

Оператор MAT MOVE предназначен для поэлементной переписи информации из массива в массив (типы массивов должны совпадать) с упорядочением информации в соответствии с массивом- локатором; массив-локатор может быть получен при реализации операторов MAT SORT или MAT MERGE.

Оператор осуществляет поэлементную перепись данных из массива источника, указанного до слова TO, в массив приемника, указанного после слова TO; массив приемника заполняется построчно, начиная с элемента, указанного в операторе.

Обычно после выполнения оператора MAT SORT массив переписывается полностью и приемник задается первым элементом. После выполнения оператора MAT MERGE на каждом этапе приемник задается элементом, соответствующим этапу.

Порядок переписи данных задается массивом-локатором, каждый элемент которого определяет положение элемента в массиве источника. Просмотр массива-локатора начинается с элемента, указанного в операторе.

Если длина элемента (поля элемента) символьного массива-источника меньше длины элемента массива-приемника, то элементы, записанные в массив приемника, имеют в конце пробелы; если длина элемента (поля элемента) символьного массива-источника больше длины элемента массива-приемника, то значение элемента усекается справа.

Для переписи некоторого числа элементов массива можно использовать параметр <числовая переменная>, являющийся счетчиком числа переписанных элементов. Значение этой переменной должно быть присвоено пользователем до реализации оператора MAT MOVE и находиться в диапазоне 1—7999 включительно. Дробная часть значения этой переменной при реализации оператора отбрасывается.

По окончании реализации оператора MAT MOVE числовой переменной (если параметр указан в операторе) автоматически присваивается значение, равное числу элементов, переписанных из массива в массив.

Выполнение оператора заканчивается, если:

исчерпан массив-локатор;

в массиве-локаторе обнаружен код HEX (0000);

заполнен массив приемника;

число переписанных элементов равно значению параметра < числовая переменная>

Пример 154.

```
10 MAT MOVE A(),L%(2)TOB(5)
20 MAT MOVE A%(),L%(1,3),MTOB%(1,2)
30 MAT MOVE P%()(3,2),L%(1)TOA2%(3)
```

Пример 155.

```
10 DIM S%(1,1)10,L%(2,5)2,R2%(10)2,M%(2,5)1,B%(1)1
20 S%(1,1)="ПРОГРАММА"
30 MAT REDIM S%(2,5)1
40 MAT SORT S%()TOR2%(),L%()
50 PRINT S%()
60 MAT MOVE S%(),L%(1,1)TOM%(1,1)
70 MAT PRINT M%;B%
ПРОГР
АММА
 ААГМ
МОПРР
```

3. СОРТИРОВКА ВТОРОГО ТИПА

Определяющим требованием, предъявляемым к сортируемому массиву, является упорядоченность значений элементов массива в строках, поэтому сортировку можно назвать слиянием рассортированных однострочных массивов. Сортировка второго типа позволяет «наращивать» значения в строках, т. е. по мере использования всех значений в строке заполнять ее новыми бóльшими значениями, которые также должны быть упорядочены.

Другой отличительной чертой этого типа сортировки является многоэтапность ее выполнения, т. е. строки массива сливаются за несколько этапов. Каждый этап сортировки второго типа аналогичен этапам сортировки первого типа и заключается в получении массива-локатора и упорядочении. Массив считается рассортированным, если при получении массива-локатора исчерпались все строки сортируемого массива.

Длина массива-локатора строго не ограничивается. Увеличивая ее, можно сократить число этапов сортировки до числа сортируемых строк.

Длина рабочего массива 2 должна быть не меньше числа сортируемых строк, а длина рабочего массива 1 — на единицу больше.

Длина элемента массива-локатора и рабочего массива 2 составляет два байта, а рабочего массива 1 — один байт.

Сортируемый массив имеет размерность не более 254×254. Результирующий массив может быть сформирован по частям, поэтому при программировании его можно объявлять меньше сортируемого, что усложняет программу.

При программировании сортировки необходимо придерживаться последовательности:

1) объявить все необходимые для сортировки массивы, в том числе:

массив-локатор, по возможности, длиннее, но не более длины сортируемого массива; рабочий массив 1 — одномерным с однобайтными элементами размерностью K + 1; рабочий массив 2 — одномерным с двухбайтными элементами размерностью не меньше K (K — число сортируемых строк);

2) оператором INIT присвоить элементам рабочего массива 1 значения HEX (01), т. е. установить счетчик элементов каждой сортируемой строки в единицу;

3) если сортируемый массив числовой, то преобразовать его оператором MAT CONVERT и получить символьный массив;

4) получить массив-локатор оператором MAT MERGE;

5) упорядочить оператором MAT MOVE часть исходного массива в соответствии с массивом-локатором, переписав ее в результирующий массив;

6) выполнить анализ на конец сортировки, т. е. проверить, последняя ли часть исходного массива переписалась при упорядочении в результирующий массив. Если переписалась не последняя часть, то сортировку нужно продолжить с п. 4, т. е. перейти к выполнению следующего этапа сортировки, сохранив значение рабочего массива 1. (Действия каждого этапа указаны в п. 4, 5, 6).

Для анализа на конец сортировки используются признаки завершения выполнения оператора MAT MERGE:

наличие HEX (0000) в первом элементе массива-локатора; признак свидетельствует о том, что все данные уже использованы на предыдущем этапе (последний элемент рабочего массива 1 имеет значение, не равное HEX (00);

наличие HEX (FF) во всех элементах рабочего массива 1, кроме последнего элемента.

Оператор MAT MERGE. Структура оператора:

MAT MERGE < метка символьного массива> TO <метка рабочего массива 1>, <метка рабочего массива 2>, <метка массива-локатора>

Оператор MAT MERGE предназначен для создания массива-локатора, используемого в операторе MAT MOVE.

Сортируемый массив должен быть символьным массивом, в котором число строк (столбцов) не более 254. Значения элементов каждой строки должны быть упорядочены по возрастанию. Одномерный массив воспринимается как один столбец, поэтому в операторе не используется.

При выполнении оператора MAT MERGE используются также массив-локатор, рабочий массив 1 и рабочий массив 2 (см. с. 186).

Перед выполнением оператора MAT MERGE элементы рабочего массива 1 должны иметь значения двоичных номеров текущих элементов строк сортируемого массива, т. е. элементов, с которых начинается сортировка. На первом этапе сортировки это первые элементы из каждой строки, а на последующих этапах — первые элементы, неиспользованные на предыдущем этапе. Строки, для которых номер текущего элемента равен HEX (FF), в сортировке не участвуют.

При выполнении оператора текущие элементы из каждой сортируемой строки сравниваются между собой. Местоположение наименьшего значения заносится в первый свободный элемент массива-локатора, а в элемент рабочего массива 1 с номером, равным номеру строки выбранного наименьшего значения, заносится номер элемента, следующего за наименьшим. Этот элемент становится текущим в строке, а выполнение оператора продолжается следующим сравнением и т. д.

Выполнение оператора заканчивается, если:

1) заполнен массив-локатор; при этом в последний элемент рабочего массива 1 заносится признак HEX (00).

2) исчерпана строка сортируемого массива;при этом номер исчерпанной строки заносится в последний элемент рабочего массива 1 (признак), а элементу с номером, равным номеру исчерпанной строки, присваивается значение HEX (FF). Если массив-локатор заполнен не полностью, то в его первый свободный элемент заносится HEX (0000).

15. Пример выполнения оператора MAT MERGE по программе 156

Номер строки массива	Начальные значения элементов массивов		Текущие значения элементов массивов на каждом из трех этапов						Примечание
	R1¤	L¤	R1¤	L¤	R1¤	L¤	R1¤	L¤	
1	01	FFFF	02	0101	02	0101	02	0101	Участвуют строки 1, 2, 3. Заканчивается выполнение оператора исчерпанием строки 3 сортируемого массива
2	01	FFFF	01	FFFF	01	0301	01	0301	
3	01	FFFF	01	FFFF	02	FFFF	FF	0302	
4	01	FFFF	01	FFFF	01	FFFF	03	0000	
1	02	FFFF	02	0201	FF	0201			Участвуют строки 1, 2. Заканчивается выполнение исчерпанием строки 1
2	01	FFFF	02	FFFF	02	0102			
3	FF	FFFF	FF	FFFF	FF	0000			
4	03	FFFF	03	FFFF	01	FFFF			
1	FF	FFFF	FF	0202					Участвует одна строка 2. Заканчивается выполнение исчерпанием строки 2
2	02	FFFF	FF	0000					
3	FF	FFFF	FF	FFFF					
4	01	FFFF	02	FFFF					

Примечание. R1¤ — рабочий массив 1; L¤ — массив-локатор четырехэлементный.

Созданный массив-локатор содержит координаты местоположения элементов части сортируемого массива в порядке возрастания значений элементов.

Возможна следующая форма записи:

10 MAT MERGE S (0) TO R1 ¤ ( ), R 2 ¤ ( ), L ¤ ( )

Проследить за выполнением оператора MAT MERGE можно на примере сортировки массива с тремя двухэлементными строками, значениями элементов которого являются одноразрядные числа:

1 6 — первая строка

4 7 — вторая срока

2 3 — третья строка

Вид чисел, преобразованных в символьный формат для сортировки, опущен. Ниже приведен один из вариантов программы, по которой выполняется сортировка массива (пример 156), а в табл. 15 — значения рабочего массива 1 (R1 ¤) и массива-локатора (L ¤) на каждом из трех этапов сортировки, т. е. для трех исполнений оператора MAT MERGE.

Для рассматриваемого примера достаточно задать длину массива-локатора, равную трем элементам.

Пример 156.

```
5 SELECT PRINT@C(65):STOP "ВКЛЮЧИТЕ ПЕЧАТЬ И НАЖМИТЕ CONTINUE"
10 DIM A(3,2),S¤(3,2)2,R1¤(4)1,R2¤(3)2,L¤(4)2,A1(6,1)
20 DATA 1,6,4,7,2,3:MAT READ A
30 MAT CONVERT A()TOS¤():REM ПРЕОБРАЗОВАНИЕ
40 INIT (01)R1¤():K=1:REM ПОДГОТОВКА РАБОЧЕГО МАССИВА 1
45 REM НАЧАЛО СОРТИРОВКИ
50 INIT (FF)L¤():MAT MERGE S¤()TOR1¤(),R2¤(),L¤()
55 PRINT /05,"МАССИВ-ЛОКАТОР ПОЛУЧЕН":PRINT
56 PRINT "R1¤   L¤"
57 FOR M=1TO4:HEXPRINT R1¤(M);:PRINT "    ";:HEXPRINT L¤(M):NEXT M
60 MAT MOVE A(),L¤(1)TOA1(K,1):REM ПЕРЕПИСЬ ЗНАЧЕНИЙ МАССИВА В РЕ
ЗУЛЬТИРУЮЩИЙ МАССИВ A1
65 PRINT "K=";K
66 MAT REDIM A1(3,2):PRINT A1():MAT REDIM A1(6,1)
67 REM АНАЛИЗ НА КОНЕЦ СОРТИРОВКИ
70 FOR M=1TO3:IF R1¤(M)<>HEX(FF)THEN90
80 NEXT M:PRINT "КОНЕЦ СОРТИРОВКИ":END
85 REM РАСЧЕТ НОМЕРА ПЕРВОГО СВОБОДНОГО В A1 ЭЛЕМЕНТА
90 IF R1¤(4)<>HEX(00)THEN110
100 K=K+4:GOTO 50
110 FOR M=1TO4:IF L¤(M)=HEX(0000)THEN130
120 NEXT M:M=M+1
130 K=K+M-1:GOTO 50
```

Строки программы с номерами, не кратными 10, необязательны. При использовании в MAT MOVE числовой переменной расчет номера свободного элемента A1 упрощается.

4. ОПЕРАТОР MAT COPY

Структура оператора:

MAT COPY [—] <метка символьного массива> [(⟨поле массива⟩)]

TO [—] < метка символьного массива> [(⟨ поле массива⟩)]

Оператор предназначен для побайтной переписи данных из массива-источника (поля массива-источника), указанного до слова TO, в массив-приемник (поле массива-приемника), указанный после слова TO. Массив рассматривается как единая цепочка символов.

Выполнение оператора MAT COPY заканчивается, если исчерпан массив-источник или массив-приемник.

Если длина массива (поля массива)-приемника меньше длины массива (поля массива)-источника, то не поместившиеся в массив (поле массива)-приемник байты отбрасываются; если длина массива (поля массива)-приемника больше длины массива (поля массива)-источника, то свободные байты в массиве (поле массива)-приемнике заполняются пробелами. Направление заполнения массива (поля массива)-приемника зависит от наличия знака «минус» (—) в операторе:

если в операторе нет знака «минус», то массив (поле массива)-приемника заполняется от левого края и байты выбираются из массива (поля массива)-источника слева направо (т. е. в том порядке, как они в нем записаны);

если в операторе перед массивом (полем массива)-источником стоит знак «минус», то байты из массива (поля массива)-источника выбираются справа налево (т. е. в порядке, обратном записанному);

если в операторе перед массивом (полем массива)-приемником стоит знак «минус», то он заполняется от правого края.

Массивом-источником и массивом-приемником может быть один и тот же массив.

Пример 157.

```
10 MAT COPY A$()TO-C$()<5,20>
20 MAT COPY L$()<7,7*2>TOL$()<1,2*2>
30 MAT COPY -M$()TOA$()
```

Пример 158.

```
10 DIM A$(1,1)10,B$(2,5)1
20 A$(1,1)="НАСТРОЕНИЕ":MAT REDIM A$(2,5)1
30 MAT COPY A$()TOB$():PRINT B$()
40 MAT COPY A$()<3,6>TOB$():PRINT B$()
50 INIT (2D)B$()
60 MAT COPY A$()<3,8>TOB$()<2,8>:PRINT B$()
70 INIT (20)B$()
80 MAT COPY A$()<3,6>TO-B$()<2,4>:PRINT B$()
```

```
НАСТР
ОЕНИЕ
СТРОЕ
Н
-СТРО
ЕНИЕ-
 ОРТС
```

5. ОПЕРАТОР MAT SEARCH

Структура оператора:

MAT SEARCH <метка символьного массива> |(<поле массива>)|,
<операция сравнения> <символьная переменная> TO
<метка массива-локатора> **STEP** [< а. в. >]
<**операция сравнения**> : = <|>|=|<=|>=|< >|

Оператор MAT SEARCH предназначен для выделения из символьного массива (поля массива) групп байтов, удовлетворяющих заданному условию относительно значения символьной переменной, указанной в операторе.

При выполнении оператора разделение массива на элементы не учитывается; просмотр байтов осуществляется слева направо по строкам массива; номера байтов, удовлетворяющих условию, заносятся слева направо в массив-локатор.

Массив-локатор должен быть символьным массивом с длиной элемента, равной 2 байт.

Оператор просматривает группы символов массива (поля массива) длиной, равной длине символьной переменной без учета оконечных пробелов (если символьная переменная заполнена только одними пробелами, то она воспринимается как один пробел).

Чтобы в длину входили и оконечные пробелы, в качестве символьной переменной следует использовать STR-функцию. Если группа символов удовлетворяет условию, заданному в операторе, то номер ее второго символа (байта) относительно начала массива (поля массива) заносится в массив-локатор в виде HEX-кода, т. е. двухразрядного шестнадцатеричного числа.

Просмотр групп символов массива (поля массива) осуществляется с шагом 1, если не задан параметр STEP <а.в.>

Если же этот параметр задан, то просмотр осуществляется с шагом, равным целой части <а.в.>, указанного после STEP, при этом $0 < |<$ а. в.$>| \leqslant 255$. При отрицательном значении шага просмотр массива (поля массива) осуществляется справа налево.

Выполнение оператора заканчивается, если:

заполнен массив-локатор;

длина группы символов, подлежащих сравнению, в конце массива меньше длины символьной переменной; в этом случае в массив-локатор, если он не заполнен полностью, заносится код HEX (0000);

просмотрен массив-источник.

Массив-локатор, составленный оператором MAT SEARCH, не может быть использован оператором MAT MOVE, так как он задает местоположение байтов (а не элементов) в массиве.

Пример 159.

```
10 MAT SEARCH A$(),=B$TOM$()
20 MAT SEARCH A$(),<STR(K$,1,3)TOR$()STEPX*5
30 MAT SEARCH P$()<Y,8>,>=K$TOP$()
```

Пример 160.

```
10 DIM P$(1)14,Q$(3,4)2
20 INIT (FF)Q$():K$="A"
30 P$(1)="BACDCBFAAFAPAR"
40 MAT SEARCH P$(),=K$TOQ$()
50 HEXPRINT Q$()

0002000000090008
000D0000FFFFFFFF
FFFFFFFFFFFFFFFF
```

В результате выполнения программы массив-локатор G ¤ ( ) содержит следующие HEX-коды:

Пример 161.

```
1 DEFFN '0"MAT SEARCH M$(),=B$TOL$ < >"
5 SELECT PRINT@C
10 DIM M$(3,2)2,B$3,L%(3,4)2
20 DATA "AB","CD","AB","CE","FA" "BC"
30 MAT READ M$
40 B$="ABC":INIT (FF)L%()
50 MAT SEARCH M$() =B$TOL$()
60 HEXPRINT L%():PRINT
70 INIT (FF)L%()
80 MAT SEARCH M$(),=B$TOL$()STEP3
90 HEXPRINT L%():PRINT

0001000500000000
FFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFF

0001000A0000FFFF
FFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFF
```

Глава XVI
ВЫВОД ГРАФИЧЕСКОЙ ИНФОРМАЦИИ

1. ОБЩИЕ СВЕДЕНИЯ

В машине «Искра 226» (исполнения 1—6) в режиме графического вывода работает БОСГИ (графический) с ФАУ 10, а в исполнениях 3, 4, 6, кроме того графопостроитель Н-306 с ФАУ 14. Графический режим вывода позволяет выполнить чертеж и вывести графические символы. Рабочим графическим полем БОСГИ является экран, имеющий линейные размеры 230×105 мм. Рабочее поле графопостроителя 200×300 мм.

Вывод графической информации выполняется пишущим узлом. Пишущим узлом БОСГИ является световой маркер — графический курсор (на экране БОСГИ одновременно могут находиться символьный и графический курсоры, когда в работе одновременно участвуют БОСГИ символьный и БОСГИ графический). Пишущим узлом графопостроителя является перо, заправляемое чернилами.

При программировании рабочее графическое поле устройств рассматривается как прямоугольная матрица из конечного числа фиксированных точек, в каждой из которых перемещающийся пишущий узел может сделать остановку. Число графических точек поля для БОСГИ составляет 256 строк×560 точек; для графопостроителя 2666 строк×4000 точек).

Началом отсчета в графическом поле является начало координат (левый нижний угол поля).

Местоположение текущей графической точки на экране БОСГИ индицируется вершиной угла графического курсора, на графопостроителе — положением пера. Местоположение текущей графической точки задается в программе целочисленными координатами в правой прямоугольной декартовой системе координат.

Вывод информации (чертежа и символов) выполняется перемещением на заданное число шагов по осям X и Y пера графопостроителя в опущенном положении и курсора в режиме вывода.

Минимальный шаг — элементарное перемещение Δ — физическая характеристика устройства, представляющая собой расстояние между соседними графическими точками по горизонтали, вертикали и диагонали. Элементарное перемещение для БОСГИ равно 0,41 мм; для графопостроителя — 0,075 мм.

Шаг вывода на графопостроитель может быть больше элементарного перемещения, если задан масштаб вывода. Отдельно задается масштаб для вывода чертежа и для вывода графического символа. В начальном состоянии масштаб для чертежной информации у БОСГИ равен 1, у графопостроителя — 6 (такие масштабы делают шаги этих устройств одинаковыми), масштаб для графического символа у БОСГИ равен 1, у графопостроителя — 2.

Установить масштаб вывода чертежа, выполняемого оператором PLOT, можно оператором PRINT, например:

```
PRINT/14, HEX (1B0208);
```

Две последние цифры аргумента HEX-функции (в примере HEX-код 08)— это значение масштаба чертежа графопостроителя; символ «;» — набираемый символ-разделитель. Если значение масштаба равно 00, то перемещения пера не будет.

Установить масштаб вывода символов можно оператором

```
PRINT/14, HEX (1B0105);
```

Две последние цифры аргумента HEX-функции (в примере HEX-код 05) — масштаб символа.

Если значение масштаба символа равно 00 или 01, то устанавливается масштаб, равный 1.

Вывод графической информации можно программировать специальным графическим оператором PLOT, а также операторами PRINT, DATA SAVE BT, ¤ GIO, для которых устройство задается оператором SELECT. Параметр <длина строки> для графических устройств означает емкость буфера вывода. Если емкость буфера недостаточна, то график будет усеченным.

Буфер вывода задается следующим образом:

```
SELECT, PLOT 10(2000) или SELECT PLOT 14 (7999)
    SELECT PRINT 10 (7999) или SELECT PRINT 14 (7999)
    SELECT TAPE 10(7999) или SELECT TAPE 14 (7999)
```

В начальном состоянии БОСГИ имеет очищенный экран, курсор в начале координат и установленный режим блокировки отображения и сохранения информации. В начальном состоянии у графопостроителя перо поднято и находится в начале координат.

2. ОПЕРАТОР PLOT

Структура оператора:

PLOT [<устройство>,] <список PLOT>

Здесь

<список PLOT>::=< элемент списка PLOT> [, <список PLOT >]

<элемент списка PLOT> : := [<а.в.>] < [<ΔX>], [<ΔY >], [<параметр>] >,

где большие угловые скобки есть символы «больше» и «меньше», а малые, как и во всех синтаксических формах, есть металингвистические скобки:

⟨**параметр**⟩ : : = { R | U | D | C | S | ⟨символьная константа⟩ | ⟨символьная переменная⟩ }

⟨**а. в.**⟩ : : = ⟨арифметическое выражение⟩, не меньше единицы;

⟨**ΔX**⟩ : : = ⟨арифметическое выражение⟩

⟨**ΔY**⟩ : : = ⟨арифметическое выражение⟩

Всегда используется целая часть арифметических выражений, которая ограничена размером (в шагах) графического поля устройства.

Параметр ΔX или ΔY задает перемещение по оси X или Y, т. е. число шагов пишущего узла по соответствующей оси.

Все целые части числовых параметров ограничены пределами $\pm$ 7999, за этими пределами возникает ошибка синтаксиса (ERR06).

Оператор PLOT предназначен для управления устройством вывода графики. Для вывода по оператору PLOT требуется буфер памяти (понятие, аналогичное длине строки печатающего устройства). По умолчанию его емкость равна 1024 байт. Максимально допустимая емкость буфера 7999. Задать емкость буфера явно можно оператором

SELECT PLOT [ФАУ] (<а.в.>)

На УВВ по оператору PLOT выводится только то, что помещается в буфере. Если емкости буфера не хватает, то следует использовать еще один оператор PLOT. После вывода по каждому оператору PLOT пишущий узел устанавливается в поднятое положение.

Перемещение пищущего узла по прямой от текущего положения задают параматрами ΔX и ΔY. Если ΔX, ΔY или оба параметра не заданы, то значение параметра по умолчанию равно нулю. Параметр ΔX за-

дает проекцию перемещения как число шагов по оси X(горизонтали) если $\Delta X > 0$, то перемещение осуществляется вправо, если$\Delta X < 0$— влево. Параметр ΔY задает аналогично перемещение либо вверх, либо вниз. Оператор PLOT заданное перемещение по прямой аппроксимирует шагами и выводит на УВВ соответствующие управляющие HEX-коды.

Перемещение пишущего узла осуществляется в поднятом положении, если задан параметр U или не задан никакой параметр, и в опущенном положении, если задан параметр D. Параметр R оператора PLOT вызывает ускоренное перемещение пишущего узла в поднятом положении в начало координат.

Параметры R, U и D действуют только в пределах одного элемента списка оператора PLOT.

Если параметр — символьная константа или переменная, то перемещение, заданное одновременно ΔX и ΔY, осуществляется в поднятом положении пишущего узла, а затем вычерчиваются символы константы или значения символьной переменной согласно таблице КОИ-8. Если задан параметр C, то ΔX задает масштаб символов. При этом ΔX должно быть положительным, а ΔY — любым. Если задан параметр S, то ΔX и ΔY задают промежуток между символами по горизонтали и вертикали соответственно. Параметры C и S действуют. пока не будет получено новое задание на один данный оператор PLOT. Если параметр C не задан, то масштаб символов равен 1. Если параметр S не задан, то промежуток между символами равен двум шагам по горизонтали.

Если в элементе списка оператора PLOT задано <a.в>,то элемент списка исполняется повторно, а число повторений задается целой частью <a.в.>, причем оно должно быть > 0, иначе — синтаксическая ошибка. Если <a. в.> не задан, то элемент списка исполняется один раз.

При программировании вывода чертежа на БОСГИ и графопостроитель следует учитывать размер рабочего поля. Выход пишущего узла за его пределы влечет за собой неправильное исполнение оператора PLOT с параметром R.

Пример 162.

```
5 PRINT HEX(03),"ДЛЯ ВЫПОЛНЕНИЯ ПРИМЕРА ПОСЛЕ КАЖДОГО STOP Н
АЖИМАЙТЕ КЛАВИШУ 'CONTINUE'"
10 PLOT 10(10,20,HEX(41)),(,,R):STOP :PRINT /10,HEX(0D)
20 PLOT 5(20,20,HEX(41)),(5,-50,U),5(10,10,HEX(43)),(,,R):ST
OP :PRINT /10,HEX(0D)
30 PLOT 10(1,1,D),(5,4,S),(7,3,"ТЕКСТ"),(2,,C),(7,3,"ТЕКСТ"),
(,,R):STOP :PRINT /10,HEX(0D)
40 PLOT (50,,),(50,,D),(-25,50,D),(-25,-50,D),(,,R):STOP :PR
INT /10,HEX(0D)
50 PLOT (100,,D),(,50,D),(-100,,D),(,-50,D),(100,50,D),(,-50
,U),(-100,50,D),(,,R):STOP :PRINT /10,HEX(0D)
```

Пример 163.

```
1 DEFFN ' 0"PLOT<"
5 PLOT <0,240,D>,<3,5,"Y">,<,,R>
6 PLOT <370,0,D>,<5,0,"X">,<,,R>
10 Y1=240:PLOT <15,Y1,U>,<0,-20,D>
20 FOR X=4TO350
30 Y2=700/(X^0.8):Y=-INT(Y1-Y2):IF Y=0THEN40
35 Y1=Y2
40 PLOT <1,Y,D>:NEXT X:PLOT <,,R>
50 PLOT <40,40,D>,<0,-40,D>,<0,40,U>,<-40,0,D>,<,,R>
60 PLOT <45,3,"33.6">,<,,R>
70 PLOT <3,45,"42">,<,,R>
80 PLOT <2,,C>,<100,100,"Y">,<3,3,HEX(3D)>>,<3,0,"700">,<3,0,
HEX(2F)>>,<3,-3,HEX(28)>>,<3,0,"X">,<2,0,HEX(5E)>>,<2,0,"0.8">,
<2,0,HEX(29)>>,<,,R>
```

Результаты выполнения PLOT представлены на рис. 10.

Текст программы показывает один из возможных способов вычерчивания оператором PLOT гиперболы по формуле

$$Y = 700/X^{0,8}.$$

Несмотря на то, что в операторе PLOT всегда автоматически используются лишь целые части его числовых параметров, в строке 30 все-таки необходим оператор Y = — INT (Y1 — Y2). Замена его на Y = — (Y1 — Y2) приведет к ошибочному результату: правая часть кривой будет вычерчена как горизонтальная прямая линия.

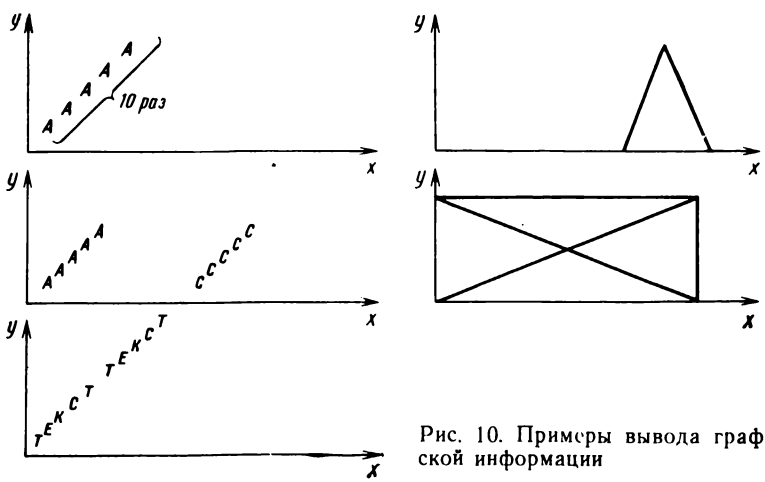


Рис. 10. Примеры вывода графической информации

3. ДОПОЛНИТЕЛЬНОЕ УПРАВЛЕНИЕ ГРАФИКОЙ

Работой устройств графического вывода можно управлять не только оператором PLOT, но и операторами PRINT и DATE SAVE BT с использованием HEX-кодов и оператором ¤ GIO, при этом точность, и скорость вычерчивания выше, а возможности программирования более глубокие, чем при использовании оператора PLOT.

Например, перемещение на 4 шага по горизонтали, т. е. по оси X, и затем на один шаг по оси Y в обратном направлении ($-Y$) можно задать оператором:

```
PRINT/14, HEX (090909090A);
```

Перемещение одновременно по осям X и Y на 3 шага можно задать:

```
SELECT PRINT 10 (7999): PRINT HEX (050505);
```

или

```
DIM C ¤ 3:INIT (05)C ¤:SELECT TAPE 14 (7999): DATA SAVE BTC ¤
```

HEX-коды, в дополнение к возможностям PLOT, позволяют стирать на экране любую часть уже вычерченного изображения, блокировать отображение чертежа на требуемое время (например, для исполнения операторов, изменяющих чертеж на экране), вновь разблокировать отображение, перемещать курсор в любую точку экрана, не изменяя изображения (в режиме СОХРАНЕНИЕ ИНФОРМАЦИИ). После включения питания машины или нажатия клавиш RESET или CLEAR устанавливаются блокировка отображения и режим СОХРАНЕНИЕ ИНФОРМАЦИИ.

Графопостроитель может самостоятельно (без управления от машины) вычерчивать графические символы таблицы КОИ-8 (ГОСТ 19768—74) по соответствующим HEX-кодам, представленным в табл. 16. Такие символы в отличие от символов, выводимых оператором PLOT, имеют высоту 16, ширину 12, промежуток между символами 7 дискрет графопостроителя (т. е. 7·0,075 мм), если масштаб символов графопостроителя установлен равным единице. После вычерчивания символа перо поднимается в правом нижнем углу матрицы символа и перемещается горизонтально на промежуток между символами. Форма начертания символов графопостроителя несколько отличается от формы символов, выводимых оператором PLOT. Масштаб символов задается по тем же правилам, что и для PLOT.

Графопостроитель помимо выполнения перемещений по шагам и вывода оператором PLOT, может самостоятельно (без управления от машины) перемещать перо в соответствии с задаваемыми (в дискретах) HEX-кодами. При этом интерполяция прямых выполняется дискретами графопостроителя более точно, чем оператором PLOT. Формат выводимой на графопостроитель информации должен быть таким:

1-й байт — HEX (1B);

2-й байт — HEX (00);

3-й байт — старшие шестнадцатеричные цифры приращения (в дискретах) по X;

16. Графические символы графопостроителя по таблице КОИ-8

Младшая цифра НЕХ-кода	Старшая шестнадцатеричная цифра НЕХ-кода															
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	ПУС		Пробел	0	@	P									Ю	П
1	НЗ	СУ1	!	1	A	Q									А	Я
2	НТ	СУ2	,,	2	B	P									Б	Р
3	КТ	СУ3	#	3	C	S									Ц	С
4	КП		¤	4	D	T									Д	Т
5	КТМ		%₀	5	E	U			НС						Е	У
6	ДА		&	6	F	V									Ф	Ж
7	ЗВ		'	7	G	W									Г	В
8	ВШ		(	8	H	X									Х	Ь
9	ГТ		)	9	I	Y									И	Ы
A	ПС		*	:	J	Z									И	З
B	ВТ	АР2	+	;	K	[									К	Ш
C	ПФ		,	<	L										Л	Э
Д	ВК		—	=	M	]									М	Щ
E	ВЫХ			>	N	^									Н	Ч
F	ВХ			?	O	—									О	0

Примечания. 1. Функции управления для НЕХ (00)—НЕХ (1F), НЕХ (85) см. в табл. 8.

2. Пустая клетка таблицы означает вычерчивание символа графопостроителя «Пробел».

4-й байт — младшие цифры приращения по X;

5-й байт — HEX (80) или HEX (00) — отрицательный или положительный знак приращения по X;

6,7, 8-й байты задают приращения по Y аналогично приращению по X.

С графопостроителя в отличие от графического БОСГИ можно вводить координаты (в дискретах графопостроителя) положения, в которое пишущий узел был перемещен операторами вывода на графопостроитель.

Оператором DATA LOAD BT задают ввод информации, имеющий следующий формат (причем ввод задают по одному байту за один раз);

1-й байт — первая слева (старшая) и вторая цифры шестнадцатеричного кода координаты X;

2-й байт — младшие цифры шестнадцатеричного кода координаты X;

3-й байт — HEX (80) или HEX (00) — отрицательный или положительный знак координаты X;

4, 5, 6-й байты передают шестнадцатеричные коды координаты Y аналогично кодам координаты X.

Пример 164.

```
5 DIM A%(6):INIT (00)A%()
10 SELECT TAPE14(7999)
20 FOR K=1TO6
30 DATA LOAD BT A%(K)
40 NEXT K
50 X=VAL(A%(2))+256*VAL(A%(1))
60 Y=VAL(A%(5))+256*VAL(A%(4))
70 IF A%(3)=HEX(00)THEN80:X=0-X
80 IF A%(6)=HEX(00)THEN90:Y=0-Y
90 PRINT "ЗНАЧЕНИЕ КООРДИНАТ В ДИСКРЕТАХ"
100 PRINT "X=";X,"Y=";Y
```

Глава XVII

ОРГАНИЗАЦИЯ ОБМЕНА С НЕСТАНДАРТНЫМИ УСТРОЙСТВАМИ ВВОДА-ВЫВОДА

1. ИСПОЛЬЗОВАНИЕ ПРОГРАММНО-ТЕХНИЧЕСКИХ СРЕДСТВ ДЛЯ РЕАЛИЗАЦИИ ОБМЕНА С УСТРОЙСТВАМИ ВВОДА-ВЫВОДА

Как уже было сказано выше, ПИД имеет двухпроцессорную структуру, т. е. содержит центральный процессор (ЦП) и канальный процессор (КП). Оба процессора принимают участие в реализации обмена информацией ПИД с устройствами ввода-вывода, однако возможна реализация обмена и непосредственно с центральным процессором, минуя канальный процессор.

ЦП выдает задание КП на обмен, а КП осуществляет непосредственную организацию обмена. Операция ввода-вывода осуществляется под управлением драйвера, т. е. специальной программы, состоящей из конкретных команд, реализующих оператор ввода-вывода, и составляющей программную часть средств реализации обмена. Драйверы всех стандартных устройств находятся в управляющей памяти канального процессора (ПЗУ КП) и реализуют операторы обмена языка БЕЙСИК, ориентированные на обмен с конкретным устройсттвом или несколькими устройствами.

Для нестандартных устройств в языке не предусмотрены специальные операторы обмена. Есть лишь обобщенный оператор обмена ¤ GIO (GENERAL INPUT — OUTPUT), для которого нет постоянного драйвера в ПЗУ. Драйвер для ¤ GIO составляется пользователем в командах канального процессора и включается в тело оператора ¤ GIO. Такой драйвер хранится в оперативной памяти в зоне программы пользователя. В одной программе пользователя может быть столько драйверов, сколько раз был применен оператор ¤ GIO. Оператор ¤ GIO можно использовать и для стандартных УВВ, но в этом нет особой необходимости.

ЦП, выдавая задание КП на обмен, указывает ему, куда следует обращаться за драйвером (в ОП или ПЗУ).

КП содержит восемь каналов связи с УВВ, по которым одновременно может производиться обмен, однако функционально к магистрали ввода-вывода, осуществляющей связь ПИД с УВВ, в каждый момент времени может быть подключено только одно УВВ.

Каждый канал КП имеет свою оперативную память — зону канала, со следующей структурой:

зона признаков — П (емкость — 4 бит); первый (младший) бит — флаг драйвера, т. е. признак, используемый по желанию пользователя, второй бит — признак прерывания, третий и четвертый биты — служебные;

зона состояний — С (4 бит); хранит информацию о состоянии выполняемой команды ввода-вывода или процедуры обмена;

зона буфера — Б (8 бит), т. е. зона информационного байта обмена; эту зону делят на две части потетрадно: БС — старшая тетрада, БМ — младшая тетрада;

зона ФАУ (8 бит);

зона адреса драйвера УВВ — АУ (16 бит); хранит текущий адрес команды драйвера УВВ;

зона адреса буфера данных — АО (16 бит);

зона длины буфера данных ОП — ДМ (16 бит); хранит текущую длину буфера данных в зоне ОП, т. е. для счетчика байтов, участвующих в обмене; первоначальное значение — длина порции обмена.

Итак, способ соединения ПИД с УВВ — магистральный через БИФ УВВ. Обмен данными между КП и БИФ происходит по магистрали ввода-вывода в строгом соответствии с интерфейсом ввода-вывода.

2. ИНТЕРФЕЙС ВВОДА-ВЫВОДА

Под *интерфейсом ввода-вывода* понимают совокупность программно-технических средств, реализующих обмен на участке ПИД — БИФ.

Чтобы программировать обмен с нестандартными устройствами с помощью оператора ¤ GIO, необходимо знать принцип работы интерфейса ввода-вывода.

Функциональными узлами интерфейса являются ПИД и БИФ (блок интерфейсный функциональный, управляющий работой конкретного УВВ, т. е. выполняющий операции, связанные с приемом и передачей данных), соединенные магистралью ввода-вывода.

Инициирующим устройством всегда является ПИД, т. е. только он выдает в БИФ команды обмена — универсальные интерфейсные команды (табл. 17), а БИФ исполняет их. Команды передаются по шинам команд. Кроме команд ПИД выдает в БИФ сигнал начальной установки (НУ) по специальной шине и сигнал ВЫЗОВ БИФ по шине ВЫЗОВ при выдаче каждой интерфейсной команды.

ПИД может выдавать байт информации по информационным шинам. Это может быть байт информации обмена, т. е. передаваемый далее БИФ в УВВ, или специальная команда данного БИФ (эта команда передается в БИФ одновременно с универсальной интерфейсной командой ПРИНЯТЬ КОМАНДУ). Специальная команда является для БИФ

17. Таблица интерфейсных команд

Интерфейсная команда	Код команды	Допустимые коды состояний (см. табл. 18)	Примечание
Установить связь (УС)	0	0, 2, 6	ФАУ передается по информационным шинам
Начальная установка (НУ)	1	0, 1, 2	
Разрешить запрос на прерывание (РЗП)	2	0, 2	Сигнал ЗАПРОС формируется после выполнения одной из команд приема либо по инициативе УВВ
Выдать состояние (ВС)	4	0, 1, 2, 3, 7	
Принять первый байт (ППБ)	8	0, 1, 2, 3, 7	
Принять байт (ПБ)	9	0, 1, 2, 3, 7	
Принять байт последний (ПБП)	A	0, 1, 2, 3, 7	
Принять команду (ПК)	B	0, 1, 2	Команда передается по информационным шинам
Выдать первый байт (ВПБ)	C	0, 1, 2, 3, 7	
Выдать байт (ВБ)	D	0, 1, 2, 3, 7	
Выдать байт последний (ВБП)	E	0, 1, 2, 3, 7	
Выдать уточненное состояние (ВУС)	F	0, 2	Команда выдается только в том случае, если есть КВУС. Уточненное состояние выдается БИФ по информационным шинам

18. Универсальные интерфейсные состояния

Интерфейсное состояние	Код	Примечание
Авария (А)	0	
Команда выполняется (КВП)	1	
Команда выполнена (КВ)	2	
Команда выполнена со сбоем (КВС)	3	
Команда выполнена, есть запрос на прерывание (КВЗП)	6	Только на команду УС
Команда выполнена, есть уточненное состояние (КВУС)	7	

управляющей, т. е. задающей режим работы БИФ (масштаб, скорость обмена и т. д.).

БИФ, отрабатывая интерфейсную команду, выдает в ПИД по информационным шинам информационный байт, полученный ранее от УВВ, или уточненное состояние (т. е. состояние специфическое для данных БИФ и УВВ). Уточненное состояние выдается только в том случае, если из ПИД поступила команда ВЫДАТЬ УТОЧНЕННОЕ СОСТОЯНИЕ.

При реализации каждой интерфейсной команды БИФ выдает универсальное интерфейсное состояние выполняемой команды (табл. 18) по шинам состояний, а по окончании приема и исполнения команды по шине ОТВЕТ выдает в ПИД сигнал ОТВЕТ. Шина ЗАПРОС используется только активными БИФ для выдачи сигнала запроса на прерывание.

Если ПИД — источник, и БИФ — приемник информации, то используются команды ПБ, ППБ, ПБП; если ПИД — приемник, а БИФ — источник информации, то используются команды ВБ, ВПБ, ВБП.

Возможны следующие режимы обмена:

однобайтный режим, который реализуется последовательностью исполняемых интерфейсных команд ПБ, ВС, ..., ВС, ПБ, ВС, ... ВС или ВБ, ВС, ..., ВС, ВБ, ВС, ..., ВС, т. е. после каждой команды обмена ПБ (ВБ) выполняется столько команд ВС, сколько потребуется для того, чтобы состояние стало КВ (см. табл. 18). Команда ВС не изменяет и не портит команду обмена и может быть выполнена многократно за время выполнения команды обмена (команда обмена выполняется дольше);

двухбайтный режим, при котором порция обмена составляет шестнадцатиразрядное двоичное число:

ППБ, ВС, ..., ВС, ПБП, ВС, ..., ВС, ...
ВПБ, ВС, ..., ВС, ВПП, ВС, ..., ВС, ...

групповой режим, при котором порция обмена содержит более 16 бит; первый байт порции выдается командой ППБ (ВПБ), все промежуточные — командой ПБ (ВБ), последний байт — командой ПБП (ВБП).

Интерфейсная команда РЗП (разрешить запрос на прерывание) переводит БИФ в активное состояние, после чего он может выставлять запрос на прерывание (если в этом БИФ есть блок, вырабатывающий сигнал ЗАПРОС), снимаемый сигналом ОТВЕТ. Если в активное состояние переведены несколько БИФ, то ЗАПРОС выставляется в соответствии с маской прерываний по приоритетности.

3. КОМАНДЫ КАНАЛЬНОГО ПРОЦЕССОРА И ИХ ИСПОЛЬЗОВАНИЕ В ОПЕРАТОРЕ ¤ GIO

Структура оператора ¤ GIO:

¤ **GIO** : := [<устройство>,] (<символьный аргумент 1>, <символьный аргумент 2>) [(<символьный аргумент 3>)]

Здесь <**символьный аргумент 1**> : := <символьная константа>| <символьная переменная> | <метка символьного массива>

<**символьный аргумент 2**> : := <символьная переменная | < метка символьного массива>

<**символьный аргумент 3**> : := <символьная переменная> | <метка символьного массива>.

Параметр <символьный аргумент 1> является драйвером; параметр <символьный аргумент 2> содержит данное, выводимое оператором в БИФ, или данное, принятое от БИФа; параметр <символьный аргумент 3> задает резерв оперативной памяти канала.

Драйвер представляет собой совокупность НЕХ-кодов команд КП (одной команде соответствует четырехразрядный шестнадцатеричный код).

Команда КП имеет переменную адресность, т. е. содержимое одного или нескольких из четырех условных полей может в команде отсутствовать. Команда КП имеет следующую структуру:

Поле 1	Поле 2	Поле 3	Поле 4
Код команды	Операнд 1	Операнд 2	Относительный адрес

В мнемонической записи команды значения полей можно разделить «*» или записать их на бланке, каждое на своем поле (табл. 19).

Каждое значение поля кодируется в соответствии с табл. 20, 21. Затем коды складываются по правилам сложения шестнадцатеричных чисел. Полученное значение записывается в драйвер.

Пример. Команда безусловного перехода

```
1 БП + 3
2 команда
3 команда
4 команда
5 команда
```

Значение поля 1 — С000, значение поля 4 — 0003. В драйвер запишется их сумма, равная С003.

Выполнится команда безусловного перехода к строке 5:

$$A = A\,K + A1 + 1 = 1 + 3 + 1 = 5.$$

БП			А1	Безусловный переход к адресу А; А=АК+А1+1	А — исполнительный адрес перехода; АК — адрес команды БП; А1 — относительный шестнадцатеричный адрес | А1 |$\leqslant 2^{15}-1$
УПС		⟨сост⟩ | X_{16} число	А1	Условный переход по состоянию. Если <операнд 2>=С, то переход к А=АК+А1+1; в противном случае к А=АК+1	АК — адрес команд УПС | А1 |$\leqslant 2^{10}-1$
УПК	БМ | БС | П | С	То же	А1	Условный переход к несравнению операндов 1 и 2 к А=АК+1+А1; в противном случае к А=АК+1	|А|1$\leqslant 2^{6}-1$
УПМ	То же	»	А1	Условный переход по выполнению логического умножения. Если N ≠ ≠ 0 — переход к А=АК+А1+1; в противном случае — к А=АК+1	То же
ОБМ	⟨Интерфейсная команда⟩	»	А1	Если <операнд 2>=С, то переход к А=АК+А1+1; в противном случае — к А=АК+1	|А1|$\leqslant 2^{3}-1$
БЯ				Буфер-Ячейка	Б — зона канала;
ЯББ				Ячейка-Буфер	Я — ячейка ОП
КБ	Б ⟨К⟩	»		Засылка константы в соответствующую зону канала	0000<К>00FF
КП	П ⟨т⟩				0000<Т>0003
КС	С ⟨Л⟩				0000<Л>000F
—ТБ			А1	Б=Б—1; если Б ≠ 0, переход к А= =АК+А1+1, в противном случае— к А=АК+1	Вычитание единицы из значения зоны Б с частотой 1 кГц | А1 |$\leqslant 2^{3}-1$
—1ДМ			А1	ДМ=ДМ—1; если ДМ ≠ 0, переход к А=АК+А1+1, в противном случае — к А=АК+1	Уменьшение счетчика ДМ на единицу

⟨Интерфейсная команда⟩::=УС | ВБ | ВБП | ВПБ | ВС | ВУС | НУ | ПБ | ПБП | ППБ | ПК | РЗП
⟨Сост⟩::=КВ | КВП | КВС | КВЗП | КВУС

20. Значения и коды полей 1, 2, 3

Поле 1		Поле 2		Поле 3	
Объект	Код	Объект	Код	Объект	Код
БП	C000	БМ	0300	Авария	0000
УПС	8000	БС	0200	КВ	0800
УПН	4080	С	0100	КВП	0400
УПМ	4000	П, УС	0000	КВС	0C00
ОБМ	0300	ВБ	00D0	КВЗП	1800
БЯ	1D60	ВБП	00E0	КВУС	1C00
ЯБ	0D70	ВПБ	00C0	X1	0400
КБ	0800	ВС	0040	X2	0800
КП	0000	ВУС	00F0	X3	0C00
КС	0400	НУ	0010	X4	1000
—ТБ	2150	ПБ	0090	X5	1400
—1ДМ	2140	ПБП	00A0	X6	1800
		ПК	00B0	X7	1C00
		ППБ	0080	X8	2000
		РЗП	0020	X9	2400
		Б<К>	<К>	XA	2800
		П<Т>	<Т>	XB	2C00
		С<Л>	<Л>	XC	3000
				XD	3400
				XE	3800
				XF	3C00
				X0	0000

Для составления драйвера необходимо соблюдать следующие требования по оформлению драйвера:

строки (команды) нумеруются с шагом 1 в шестнадцатеричной системе счисления;

последними исполняемыми командами драйвера должны быть две команды выхода (типа END в БЕЙСИКЕ):

КС — засылка константы состояний в зону С канала КП;

КП — засылка константы признака в зону П канала КП;

21. Значения и коды поля 4

Поле 4		Примечание
Объект	Код	
A1	A1	Если A1≥0
A1	|A1|+0008	Если A1<0; для объектов поля 1: ОБМ, —ТБ, —1ДМ
A1	|A1|+0040	Если A1<0; для объектов поля 1: УПН, УПМ
A1	|A1|+2000	Если A1<0; для объектов поля 1: БП

22. Драйвер вывода текста на БОСГИ

Адрес A(16)	Обозначение команды	Операнд 1 (условие)	Операнд 2 (условие)	Относительный адрес (16)	HEX-код команды	Комментарий
0	КП	П0			0000	Резерв
1	ОБМ	ВС	КВП	—1	0749	Опрос состояния. Опрос повторяется, пока состояние равно КВП
2	УПС		АВАР	+E	800E	При аварии переход к команде 11
3	КБ	Б11			0811	Задержка на 17 тактов
4	—ТБ			—1	2159	
5	ЯБ				0D70	Засылка текущего байта в зону Б
6	ОБМ	ПБ	КВ	+6	0B96	Вывод байта из зоны Б канала в БИФ
7	УПС		АВАР	+9	8009	Анализ состояния. При аварии переход к команде 11
8	УПС		КВП	+1	8401	Анализ состояния. При состоянии КВП переход к команде A
9	БП			+3	C003	Переход на изменение счетчика байтов к команде D
A	ОБМ	ВС	КВ	+2	0B42	Опрос состояния. При состоянии КВ переход к команде D
B	БП			—5	E005	Переход (возврат) к команде 7
C	КП				0000	Резерв
D	—1ДМ			+2	2142	Изменение счетчика байтов
E	КС	C0			0400	Нормальное завершение обмена (вывода).
F	КП	П2			0002	Выход из драйвера
10	БП			—E	E00E	Переход к команде 3
11	ОБМ	НУ			0310	Начальная установка устройства
12	КБ	БFF			08FF	Задержка на 256 тактов
13	—ТБ			—1	2159	
14	КБ	Б08			0808	Засылка кода уточненного состояния в зону Б
15	КС	C1			0401	Выход из драйвера по аварии
16	КП	П2			0002	

Например:

1) КС С0 — обмен завершен успешно
   КС П3 — (флаг и признак прерывания)
2) КС С8 — обмен завершен, были сбои
   КП П2 — (признак прерывания)

В табл. 22 приведен пример составления драйвера вывода текста на БОСГИ.

Рассмотренный драйвер используется в следующей программе:

```
10 DIM E%4,N%4,L%23,M%46
20 ON ERROR E%,N%GOTO70
30 SELECT CO0C:SELECT PRINT05(80)
40 L%="ПРОВЕРКА ОПЕРАТОРА %GIO "
50 M%=HEX(00000749000E0811215900700896800984O1C0030042E00500
00214204000002E00E031008FF2159000004010002)
60 %GIO /05,(M%,L%)S%:GOTO 80
70 PRINT "ОШИБ";E%,"В СТРОКЕ";N%
80 SELECT CO05(80)
```

ПРИЛОЖЕНИЕ

Отличие языка БЕЙСИК 02 от языка БЕЙСИК 01

Язык БЕЙСИК 02 отличается от языка БЕЙСИК 01 возможностями некоторых операторов и составом операторов.

Операторы, выполнение которых изменилось:

1) PRINT — обеспечивает вывод в естественной форме значений, находящихся в пределах 1E — 12 $\leqslant$ | X | $\leqslant$ 9. 999999999999+12.

2) CLEAR и LOAD — исключена индикация ошибки при задании несуществующих строк.

3) LOAD — допускается использование переменной, не объявленной оператором COM при загрузке программы из символьного аргумента.

4) KEYIN — вводит измененные коды клавиш.

5) LINPUT — по следующим параметрам:

область перемещения курсора ограничена длиной редактируемого поля;

обеспечена возможность не высвечивать символ «*» перед редактируемым текстом;

при нажатии на клавишу INSERT вставляется пробел по текущему положению курсора и исключается последний символ редактируемого поля;

при нажатии на клавишу DELETE стирается символ по текущему положению курсора и вставляется пробел на месте последнего символа редактируемого поля;

при нажатии на клавишу ERASE заменяются символы в поле редактирования, начиная с текущего положения курсора, пробелами;

при нажатии на клавишу LINE ERASE заменяются символы в поле редактирования пробелами, курсор устанавливается на первое знакомство поля редактирования, т. е. в начало логической строки экрана. Символ * заменяется на пробел.

6. ¤ TRAN, ¤ PACK, ¤ UNPACK, MAT COPY, MAT SEARCH — исключена конструкция <поле массива>.

7. SELECT DISK — исключена конструкция <длина строки> и указанный адрес устройства заносится в нулевую строку таблицы устройств.

8. SELECT# — заносит указанный адрес устройства в соответствующую строку таблицы устройств, сбрасывая при этом предыдущую информацию в строке.

9. ONERROR — в строке непосредственного счета игнорируется.

10. LIST — исключена конструкция <номер строки>.

При указании логического номера (#) в операторах ввода-вывода выбор физического адреса устройства производится из строки таблицы устройств, соответствующей указанному номеру.

При отсутствии указания устройства в дисковых операторах физический адрес устройства выбирается из нулевой строки таблицы устройств.

В дисковых операторах режима адресации секторов может отсутствовать переменная, в которую заносится очередной номер сектора.

Дополнительные операторы:

1) REPLACE — обеспечивает контекстное изменение символьных переменных.

2) ASMB — предназначен для создания программ пользователя на уровне системы инструкций и организации счета по созданной программе.

Операторы, функции которых расширены, сведены в таблицу 23. Кроме отличия языка БЕЙСИК 02 от языка БЕЙСИК 01 необходимо учитывать следующие особенности БЕЙСИК-системы.

Таблица 23

Номер по пор.	Оператор	Дополнительные функции оператора
1	DIM, COM	Использование массивов, содержащих 64 000 элементов
2	COPY	Использование третьего параметра, указывающего номер начального сектора диска копии
3	LOAD, LOAD DC, LOAD DA	Использование третьего параметра — номера строки, с которого необходимо начать счет по программе
4	MAT SEARCH	Использование операции отождествления и диапазона
5	ROTATE	Циклический сдвиг вправо битов каждого байта символьной переменной
6	VAL	Возможность преобразования двух байтов в числовой формат
7	BIN	Преобразование целой части арифметического выражения в двоичное число, присваиваемое первым двум байтам символьной переменной
8	RETURN CLEAR	Использование параметра ALL, обеспечивающего сброс всего магазина возвратов из цикла и подпрограмм
9	LIMITS	Использование физического адреса устройства и четвертой числовой переменной, в которую заносится состояние файла (форма оператора с именем файла)
10	STOP	Индикация номера программной строки, содержащей оператор STOP
11	IF THEN	Использование логических выражений, содержащих логические операции AND, OR, XOR
12	RESTORE	Указание номера строки программы, содержащей оператор DATA
13	LIST	Вывод программных строк, содержащих указанный контекст, вывод таблицы номеров файлов и указания параметра
14	MOVE	Указание второго адреса устройства и перепись файлов по одному с одного диска на другой
15	HEXPRINT	Использование символьной константы
16	INIT и функции NUM, LEN, POS	Использование метки символьного массива и символьной константы в конструкции оператора
17	ADD, AND, OR, XOR, BOOL	Использование метки символьного массива в первом элементе; символьной константы и метки символьного массива во втором элементе
18	DATA	Использование в конструкции элемента DATA числового элемента и символьного операнда
19	PRINTUSING	Использование формата PU в виде метки символьного массива

Продолжение табл. 23

Номер по пор.	Оператор	Дополнительные функции оператора
20	DATA LOAD DC OPEN, DATA SAVE DC OPEN	Использование в конструкции оператора ФАУ, который заносится в нулевую строку таблицы устройств
21	ONERROR	Указание в качестве параметра перехода GOSUB THEN
22	UNPACK, ¤ UNPACK	Проверка распаковываемых тетрад на наличие цифр
23	INPUT, LINPUT	Останов программы при нажатии на клавишу HALT/STEP после выполнения этих операторов
24	TRACE	Вывод служебных символов в виде точек и вывод сообщений при выполнении оператора NEXT

1. Изменена индикация готовности системы и сбойных ситуаций в системе.
2. Программы, оттранслированные и записанные на носитель по версии системы БЕЙСИК 01, не могут использоваться при работе с версией системы БЕЙСИК 02 (все оттранслированные программы необходимо перетранслировать).
3. Добавлена возможность объединения программных строк в одну строку в процессе редактирования.

СПИСОК ЛИТЕРАТУРЫ

1. **Кетков Ю. Л.** Программирование на БЕЙСИКе.— М.: Статистика, 1978. 157 с.

2. **Курош А. Г.** Курс высшей алгебры. — М.: Физматгиз, 1963. 432 с.

3. **Семик Б. П., Монцибович Б. Р.** Программирование на языке БЕЙСИК для СМ-4. — М.: Финансы и статистика, 1982. 246 с.

4. **Экхауз Р., Моррис Л.** Мини-ЭВМ: Организация и программирование.— — М.: Финансы и статистика, 1983.

ОГЛАВЛЕНИЕ

УЧЕБНОЕ ПОСОБИЕ

Любовь Николаевна Маркелова

ЭКСПЛУАТАЦИЯ ПРОГРАММОУПРАВЛЯЕМОЙ ВЫЧИСЛИТЕЛЬНОЙ МАШИНЫ «ИСКРА 226»

Редактор **Е. В. Григорин-Рябова**
Художественный редактор **С. Н. Голубев**
Обложка художника **В. Д. Епанешникова**
Технический редактор **Н. М. Харитонова**
Корректоры **Н. Г. Богомолова, О. Ю. Садыкова**

ИБ № 4962

Сдано в набор 21.01.87. Подписано в печать 29.05.87. Т-11990.
Формат $60\times88^{1}/_{16}$. Бумага офсетная № 2. Гарнитура литературная.
Печать офсетная. Усл. печ. л. 13,72. Усл. кр.-отт. 13,97. Уч.-изд. л. 14,46.
Тираж 58 600 экз. Заказ 2184. Цена 35 к.

Ордена Трудового Красного Знамени издательство «Машиностроение».
107076 Москва, Стромынский пер., 4

Московская типография № 4 Союзполиграфпрома
при Государственном комитете СССР
по делам издательств, полиграфии и книжной торговли.
129041, Москва, Б. Переяславская, 46

УВАЖАЕМЫЙ ЧИТАТЕЛЬ!

Издательство «Машиностроение» выпустит следующие книги по электронной вычислительной технике:

1987 год

Научно-популярная литература

Александров В. В., Шнейдеров В. С. **Рисунок, чертеж, картина на ЭВМ** / Под общ. ред. В. В. Александрова. — Л.: Машиностроение, 1987. — 7,5 л.: ил. — (Научно-популярная библиотека школьника). — (В обл.): 35 к.

В популярной форме рассказано об электронно-вычислительных машинах, которые умеют выполнять чертежи, рисовать и генерировать цветные изображения. Кратко описано их устройство. Показано, как можно просто дать задание ЭВМ и использовать ее в качестве инструмента конструктора, проектировщика, инженера и специалистов других профилей. Приведены примеры использования «электронных художников» на производстве, в архитектуре, в искусстве. В приложении представлены репродукции небольшой выставки графических работ, выполненных ЭВМ как по заказу человека, так и самостоятельно.

Изложение построено на базе материала программы средней школы.

Для школьников, учащихся ПТУ и техникумов.

Производственная литература

Прохоров А. Ф. **Конструктор и ЭВМ.** — М.: Машиностроение, 1987. — 15 л.: ил. (В пер.): 1 р. 20 к.

Рассмотрены вопросы взаимодействия конструктора и ЭВМ в процессе разработки и эксплуатации систем автоматизированного проектирования (САПР). Предоставлены методы формилизации проектно-конструкторских задач и их решения с помощью средств вычислительной техники.

Для инженерно-технических работников, занимающихся или интересующихся вопросами автоматизации проектирования и конструирования в различных отраслях промышленности.

Электронно-вычислительная машина ЕС-1035: Пер. с болг./Под ред. Ж. Й. Железова. — М.: Машиностроение, 1987. — 21 л.: ил. — Пер. изд.: Електронно-изчислителна машина ЕС-1035/Под ред. Ж. Й. Железова (София, НРБ, 1985). — (В пер.) 1 р. 80 к.

В книге болгарских авторов рассмотрены принципы работы и особенности архитектуры ЭВМ ЕС-1035 ряда 2. Изложены структура и организация управления процессора, выполнение операций

ввода-вывода, конструкция и функционирование внутренних и внешних запоминающих устройств, управление всей системой с помощью пульта оператора. Особое внимание уделено накопителям на магнитных дисках и магнитной ленте, новейшим внешним ЗУ, обработке массивов информации с помощью матричного процессора.

Учебная литература

Для вузов

Савета Н. Н. **Периферийные устройства ЭВМ:** Учеб. для студентов вузов. — М.: Машиностроение, 1987. — 25 л.: ил. — (В пер.): 1 р. 20 к.

Приведены основные сведения о периферийных устройствах ЭВМ и об организации их взаимодействия с моделями ЕС ЭВМ. Основное внимание уделено методам и средствам подготовки данных для ЭВМ, ввода в нее различной информации, вывода, отображения и регистрации результатов машинной обработки информации в различных видах и формах. По каждому классу периферийных средств на примерах современных моделей рассмотрены их устройство, принцип работы, характерные признаки и особенности. Указаны перспективы развития периферийной техники.

Для ПТУ

Катаев Е. А. **Конструкция, техобслуживание и ремонт ЭБМ «Искра 2106» и ЭБТ «Нева 501»:** Учеб. пособие для подготовки механиков по обслуживанию и ремонту вычислительных машин. — М.: Машиностроение, 1987. — 24 л.: ил. — (В пер.): 85 к.

Приведены технико-эксплуатационные характеристики, особенности программирования, основные устройства и их взаимодействие электронной бухгалтерской машины «Искра 2106» и электронного бухгалтерского табулятора «Нева 501». Описана работа электронных схем, рассмотрены характерные неисправности узлов машин с анализом причин их появления и методикой поиска. Книга может быть полезна работникам ремонтных заводов, мастерских, вычислительных центров, машиносчетных станций и бюро при наладке, техническом обслуживании и ремонте указанных машин.

1988 год

Производственная литература

Воробьев В. И. **Математическое обеспечение ЭВМ в науке и производстве.** — Л.: Машиностроение, 1988. — 10,5 л.: ил. — (ЭВМ в производстве). — (В обл.): 60 к.

Рассмотрены вопросы разработки, применения и эксплуатации математического обеспечения ЭВМ для автоматизации производства и научных исследований. Даны классификация и структуры языков программирования и рекомендации по их выбору в конкретных задачах. Приведены примеры организации структурных данных и процедур их обработки для проверки правильности программ. Описаны основные принципы построения программного обеспечения локальных и региональных сетей ЭВМ.

Для инженерно-технических работников и руководителей подразделений предприятий, не имеющих специальной подготовки в области информатики.

Майоров С. А., Кириллов В. В., Приблуда А. А. **Введение в микроЭВМ.** — Л.: Машиностроение, 1988. — 25 л.: ил. — (В пер.): 1 р. 80 к.

Книга в многокрасочном исполнении подготовлена как практическое руководство для инженеров и техников, не имеющих систематической подготовки в области микроЭВМ. Она охватывает практически все ключевые вопросы функционирования и использования микроЭВМ. Рассмотрены отечественные и зарубежные структуры микроЭВМ, специфика их работы и использования, взаимодействие с различными внешними устройствами. Даны методика построения моделей микроЭВМ и рекомендации по организации «практической работы» с рассматриваемыми микропроцессорами и микроЭВМ.

Для инженерно-технических работников, занимающихся созданием и эксплуатацией технических устройств с микроЭВМ.

Ханенко В. И. **Информационные системы.** — Л.: Машиностроение, 1988. — 7,5 л.: ил. — (ЭВМ в производстве). — (В обл.): 45 к.

Рассмотрены вопросы создания и развития информационных систем, проблемно-ориентированных на решение задач интенсификации цикла исследование — проектирование — подготовка производства — производство. Изложены вопросы эффективного использования информационных ресурсов, совершенствования информационных потоков, проектирования баз данных. Рассмотрены вопросы построения систем поддержки задач управления научно-техническим развитием на уровнях предприятие — отраслевой комплекс — регион.

Для инженерно-технических работников проектных и исследовательских организаций, не имеющих специальной подготовки в области информатики и автоматизации производства.

Учебная литература

Для повышения квалификации

Кичев Г. Г., Некрасов Л. П. **Архитектура и аппаратные средства мини-ЭВМ СМ-1600:** Учеб. пособие для подготовки и по-

вышения квалификации кадров в учебной сети ЦСУ СССР. — М.: Машиностроение, 1988. — 30 л.: ил. (В пер.): 1 р.

Приведены основные технические характеристики мини-ЭВМ СМ-1600, система команд и примеры их использования. Дано описание процессоров и каналов внешних устройств. Рассмотрены структура и работа комплекса, средства диагностирования.

Может быть использовано специалистами по эксплуатации и обслуживанию комплекса аппаратных средств мини-ЭВМ СМ-1600 в НИИ, КБ и вычислительных центрах различных отраслей народного хозяйства.

35 коп.

‹ МАШИНОСТРОЕНИЕ ›