

Справочное пособие по
программированию на алгоритмическом
языке

БЕЖСИК

для бытовых микрокомпьютеров класса

ZX SPECTRUM

10 PRINT "country"; TAB 10; "city"; TAB 24 "in

20 READ C\$, S\$, N

30 DATA "RUSSIA", "Rostov-on-Don", 344000

40 PRINT C\$; TAB 10; S\$; TAB 24; N

. RUN

city

index
344000

СПРАВОЧНОЕ ПОСОБИЕ
ПО ПРОГРАММИРОВАНИЮ
НА АЛГОРИТМИЧЕСКОМ ЯЗЫКЕ
БЕЙСИК
ДЛЯ БЫТОВЫХ МИКРОКОМПЬЮТЕРОВ
КЛАССА ZX SPECTRUM

ББК 32.85

Составители: РОЗИН Лев Григорьевич
НИКОЛАЕВ Сергей Юрьевич

Розин Л.Г., Николаев С.Ю.

Бейсик: (справочное пособие по программированию на алгоритмическом языке для бытовых микрокомпьютеров класса ZX SPECTRUM). — Ростов-на-Дону: МП "Орфей", ТОО "Икар", 1992. — 84 с.

ISBN 5-87807-008.1

МП "Орфей", 1992 г.

© ТОО "Икар", 1992 г.

ВВЕДЕНИЕ	6
1. СВЕДЕНИЯ ОБ ИСПОЛЬЗОВАНИИ	
1.1. Запуск Бейсика.	7
1.2. Клавиатура.	7
1.3. Встроенный редактор Бейсика.	10
1.4. Запуск программы на Байсике.	12
1.5. Загрузка и выгрузка Бейсик-программы.	13
2. ОБЩИЕ СВЕДЕНИЯ О ПРОГРАММИРОВАНИИ НА БЕЙСИКЕ.	
2.1. Структура программы.	15
2.2. Алфавит.	15
2.3. Константы.	19
2.4. Переменные.	19
2.5. Массивы.	20
2.6. Выражения.	21
2.7. Порядок выполнения операций.	23
2.8. Выражения над строками символов.	23
2.9. Функции.	24
2.10. Работа с экраном.	25
3. ОПЕРАТОРЫ, ФУНКЦИИ И КОМАНДЫ.	
3.1. Общие сведения.	27
3.2. Функция ABS.	27
3.3. Функция ACS.	28
3.4. Функция ASN.	28
3.5. Функция ATN.	29
3.6. Функция ATTR.	29
3.7. Оператор BEEP.	30
3.8. Функция BIN.	30
3.9. Оператор BORDER.	30
3.10. Оператор BRIGHT.	30

3.11. Команда CAT.	31
3.12. Функция CHR\$.	31
3.13. Оператор CIRCLE.	32
3.14. Оператор CLEAR.	33
3.15. Оператор CLOSE#.	33
3.16. Оператор CLS.	33
3.17. Функция CODE.	34
3.18. Команда CONTINUE.	34
3.19. Команда COPY.	34
3.20. Функция COS.	35
3.21. Оператор DATA.	35
3.22. Оператор DEF FN.	35
3.23. Оператор DIM.	36
3.24. Оператор DRAW.	37
3.25. Оператор ERASE.	38
3.26. Функция EXP.	38
3.27. Оператор FLASH.	38
3.28. Функция FN.	39
3.29. Операторы FOR и NEXT.	39
3.30. Оператор FORMAT.	40
3.31. Операторы GO SUB и RETURN.	40
3.32. Оператор GO TO.	41
3.33. Оператор IF.	42
3.34. Функция IN.	43
3.35. Оператор INK.	44
3.36. Функция 'NKEY\$.	45
3.37. Оператор INPUT.	45
3.38. Функция INT.	46
3.39. Оператор INVERSE.	47
3.40. Функция LEN.	47
3.41. Оператор LET.	48
3.42. Команда LIST.	48
3.43. Команда LLIST.	49
3.44. Функция LN.	49
3.45. Команда LOAD.	50
3.46. Команда LPRINT.	51
3.47. Команда MERGE.	51
3.48. Оператор MOVE.	51
3.49. Команда NEW.	52

3.50. Оператор OPEN#.	52
3.51. Оператор OUT.	53
3.52. Оператор OVER.	54
3.53. Оператор PAPER.	54
3.54. Оператор PAUSE.	55
3.55. Функция PEEK.	56
3.56. Функция PI.	56
3.57. Оператор PLOT.	57
3.58. Функция POINT.	57
3.59. Оператор POKE.	58
3.60. Оператор PRINT.	58
3.61. Оператор RANDOMIZE.	60
3.62. Оператор READ.	61
3.63. Оператор REM.	62
3.64. Оператор RESTORE.	62
3.65. Оператор RETURN.	63
3.66. Функция RND.	63
3.67. Команда RUN	64
3.68. Команда SAVE.	64
3.69. Функция SCREEN\$.	65
3.70. Функция SGN.	65
3.71. Функция SIN.	66
3.72. Функция SQR.	66
3.73. Оператор STOP.	67
3.74. Функция STR\$.	67
3.75. Функция TAN.	68
3.76. Функция USR.	68
3.77. Функция VAL.	70
3.78. Функция VAL\$.	70
3.79. Команда VERIFY.	71
Приложение 1. Полный набор символов Спектрума.	72
Приложение 2. Сообщения интерпретатора Бейсика.	80
Приложение 3. Примеры программ.	81

Бейсик — широко распространенный язык программирования, разработанный в 1965 г. в Дартмутском колледже (США) по заказу фирмы General Electric. Название языка является аббревиатурой английских слов Beginner's All-purpose Symbolic Instruction Code (многоцелевой язык символических команд для начинающих). Язык Бейсик был задуман как простой и удобный для пользователей язык, позволяющий программировать большой круг задач. В языке использованы обозначения, подобные обычным математическим.

В настоящем пособии описан Бейсик, реализованный встроенным интерпретатором бытовых компьютеров класса ZX — Spectrum. Пособие состоит из трех разделов. В первом разделе даны необходимые сведения о запуске интерпретатора, работе с клавиатурой компьютера, структуре памяти, способах редактирования, записи и считывания готовых программ с магнитофона. Во втором разделе даны общие сведения о программировании на Бейсике. Синтаксис и семантика всех команд, операторов и функций Бейсика описаны в разделе 3. В приложении приведены полный набор символов «Спектрума» и сообщения интерпретатора. Использование понятий и возможностей языка иллюстрируется по ходу изложения на примерах.

Для пояснения положений синтаксиса языка в данном пособии используются следующие условные обозначения:

а) элементы формата, представленные русскими словами (например, переменная, выражение и т.д.), должны указываться пользователем при написании программы согласно правилам, приведенным в тексте. Элементы формата, представленные английскими словами (например, LET, GO TO, STOP и т.д.), должны указываться такими, как они записаны в формате, поскольку они входят в словарь языка Бейсик;

б) сокращение <нс>, записанное в формате перед оператором, означает «произвольный номер строки»;

в) угловыми скобками отмечены неотъемные элементы оператора или команды, которые записывает пользователь. Например:

<нс> FOR <переменная> = <выражение> TO <выражение>

г) квадратные скобки указывают на необязательный выбор одного элемента из нескольких возможных. Например:

<нс> FOR <переменная> = <выражение> TO <выражение> [STEP <выражение>]

1. СВЕДЕНИЯ ОБ ИСПОЛЬЗОВАНИИ.

1.1. Запуск Бейсика.

Программа интерпретатора Бейсика находится в ПЗУ компьютера. Запуск программы происходит автоматически после включения питания. Сообщение на экране телевизора: "© 1982 Sinclair Research Ltd" — указывает на то, что компьютер готов к вводу и выводу программы или выполнению команд Бейсика. Перезапуск системы может быть выполнен повторным выключением и включением питания, либо выполнением команды Бейсика NEW. На некоторых компьютерах имеется специальная кнопка "Сброс", с помощью которой также можно осуществить перезапуск системы.

1.2. Клавиатура.

Клавиатура Спектрума имеет 40 клавишей. На клавишах написаны буквы и цифры, ключевые слова Бейсика, название функций, знаки пунктуации, а также дополнительные символы, такие как скобки, процент и т.д. Ключевые слова и названия функций в Спектруме вводятся нажатием одной клавиши, а не побуквенно. Большинство клавишей имеет пять значений. Определенное значение каждой клавиши выбирается с помощью функциональных клавишей SYMBOL SHIFT (далее будет обозначаться как SS) или CAPS SHIFT (CS).

Рассмотрим режимы работы клавиатуры на примере клавиши "P". Обозначим каждое из значений клавиши цифрами от 1 до 5. Рисунок клавиши приведен на рис. 1.

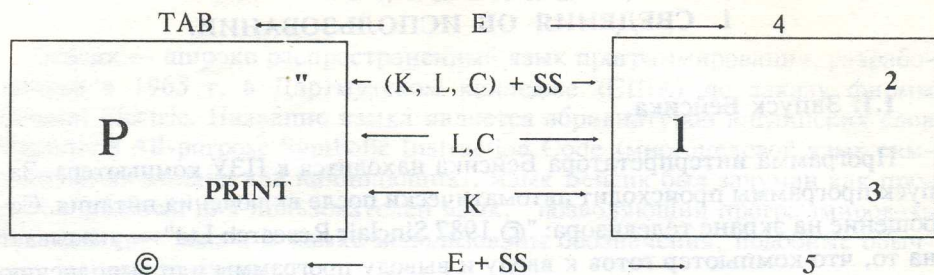


Рис. 1. Клавиша с пятью функциями.

После включения компьютера нажмите на клавишу ENTER или SPACE. В левом нижнем углу экрана появится мигающий курсор с буквой «К», указывающий, в каком именно месте будет введен знак с клавиатуры. В режиме К вводятся все команды языка Бейсика, написанные на клавише в месте, обозначенном цифрой 3 на рис.1. Это относится к трем нижним рядам клавиатуры. В случае клавиши Р — это инструкция PRINT. В верхнем ряду клавиш Спектрума в режиме К печатаются цифры.

После введения ключевого слова режим работы клавиатуры автоматически изменяется на L. В этом режиме вводятся все прописные буквы, обозначенные на рис. 1 цифрой 1. Для ввода строчных букв нужно одновременно с клавишей нажать на CAPS SHIFT. Переключение с режима L на режим C (постоянно заглавные буквы) производится одновременным нажатием на CS и 2, при этом на экране мигающий курсор L меняется на C. Возвращение в режим L возможно после повторного нажатия на CS и 2. В режимах К, L, C ввести символ, обозначенный цифрой 2 на рис. 1, можно одновременным нажатием на SIMBOL SHIFT и данную клавишу. В случае клавиши Р это будет символ (") — двойные кавычки.

Следующим режимом работы Спектрума является режим Е, т.е. расширенный режим. Он включается одновременным нажатием клавиш SS и CS. Текущий курсор при этом изменяется на Е. При работе в режиме Е вводятся символы, расположенные над клавишей (позиция 4 на рис. 1). После введения символа режим автоматически меняется на L или C.

Символы, расположенные под клавишей (позиция 5 на рис. 1), вводятся в режиме Е при одновременном нажатии на клавишу SS.

Проиллюстрируем выше сказанное простым примером, выполнив в непосредственном режиме команду:

PRINT "P"; TAB 5; "©" ENTER

В ответ компьютер выведет в верхней строчке экрана в первой позиции символ P и в шестой позиции символ ©.

Работа с верхним рядом клавиш несколько отличается от работы с клавишами нижних рядов. Клавиши верхнего ряда могут иметь до шести значений. Рассмотрим их на примере клавиши с цифрой 1 (рис. 2).

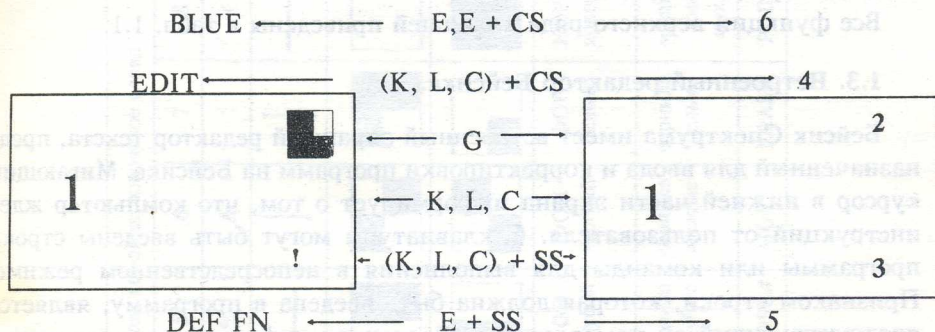


Рис. 2.

Символы, обозначенные в позиции 1, вводятся в режимах K, L, C.

Символы, обозначенные в позиции 2 (клавиши от 1 до 8), вводятся в графическом режиме, который включается одновременным нажатием клавиш CS и 9. При этом на экране появляется мигающий курсор G. Выход из графического режима происходит после одновременного нажатия на CS и 9, либо только нажатия на клавишу 9. Инверсный режим ввода графических символов можно получить одновременным нажатием на CS (или SS) и соответствующую клавишу.

В графическом режиме можно также с помощью клавиш от A до U вводить символы, графика которых определена пользователем (п 3.76).

Символы, обозначенные в позиции 3, вводятся в режимах K, L, C одновременным нажатием на SS и соответствующую клавишу.

В позиции 4 на рис. 2 находятся команды Спектрума, для выполнения которых необходимо одновременно с клавишей нажать на CS.

Ключевые слова в позиции 5 вводятся в режиме E при одновременном нажатии на клавишу SS.

Команды в позиции 6 верхнего ряда клавиш позволяют определить цвет

фона и цвет чернил в процессе ввода в память компьютера программы, написанной на Бейсике (без использования команд INK и PAPER), а также задавать и отменять режимы повышенной яркости и мерцания. Команды позиции 6 вводятся в режиме E, либо в режиме E+CS. Эти команды можно вводить в любом месте после введения номера строки. Действие команды распространяется со знакоместа, перед которым была введена команда, на весь последующий введенный текст до отмены режима для команд мерцания и повышенной яркости, либо до задания другого цвета для команд, определяющих цвета.

Все функции верхнего ряда клавишей приведены в табл. 1.1.

1.3. Встроенный редактор Бейсика.

Бейсик Спектрума имеет встроенный экранный редактор текста, предназначенный для ввода и корректировки программ на Бейсике. Мигающий курсор в нижней части экрана информирует о том, что компьютер ждет инструкций от пользователя. С клавиатуры могут быть введены строки программы или команды для выполнения в непосредственном режиме. Признаком строки, которая должна быть введена в программу, является предшествующий ей номер строки. Введенная информация появляется в нижней части экрана, а курсор перемещается вправо по мере ввода очередных знаков. Если строка не помещается в одной строке экрана, то она перемещается вверх, а текст вводится ниже. Каждая введенная строка заканчивается нажатием на клавишу ENTER. После нажатия на ENTER интерпретатор проверяет правильность вводимой строки и, если нет ошибок, вводит ее в память, высвечивая в верхней части экрана, или выполняет, если строка не имеет номера.

Если в строке содержится ошибка, то в месте обнаружения первой ошибки появляется дополнительный мигающий курсор "?". Строка не переносится в верхнюю часть экрана и не выполняется. Отредактировать строку можно, переместив курсор в любое место строки с помощью команд "←" (CS и 5) и "→" (CS и 8). Можно также удалить отдельный символ или ключевое слово, стоящие слева от курсора, нажав клавишу DELETE (CS и 0). После того, как строка полностью отредактирована, нужно нажать на клавишу ENTER.

Строки программы, введенные в компьютер, высвечиваются в верхней части экрана. В одной из них (введенной последней) высвечивается (сразу за номером строки) символ ">". Это, так называемый, курсор программы. Курсор программы можно перемещать с помощью команд "↑" (CS и 6) на

Таблица 1.1.

Режим	Дополн.	Вводимый символ, команда, ключевое слово									
		1	2	3	4	5	6	7	8	9	0
K, L, C											
	SS	!	@	#	\$	%	&	,	(	)	—
G	CS	EDIT	CAPS LOCK	TRUE VIDEO	INVERSE VIDEO	←	↓	↑	→	граф. реж. вкл.	DELETE
										граф. реж. выкл.	DELETE
	CS, SS									граф. реж. выкл.	DELETE
E		ФОН синий	ФОН красный	ФОН пурпурный	ФОН зеленый	ФОН голубой	ФОН желтый	ФОН белый	НОРМ. яркость	ПО-ВЫШ. яркость	ФОН черный
	CS	чернила синие	чернила красные	чернила пурпурные	чернила зеленые	чернила голубые	чернила желтые	чернила белые	мерцание выкл.	мерцание вкл.	чернила черные
	SS	DEF FN	FN	LINE	OPEN#	CLOSE#	MOVE	ERASE	POINT	CAT	FORMAT

следующую строку или на предыдущую командой "↑" (CS и 7). Строка, на которую указывает курсор программы, может быть перенесена для редактирования в нижнюю часть экрана командой EDIT (CS и 1), где ее можно редактировать, как при введении новой строки. Нажатие на ENTER приведет к запоминанию исправленной строки. Для удаления строки из программы необходимо ввести номер строки и ENTER. Если вводится строка с номером, который уже имеется в программе, то новая строка заменит собой старую.

При вводе программы возможно использование команд задания цвета фона и чернил, повышенной яркости, мерцания, инверсного режима высвечивания (верхний ряд клавишей в режиме E) для более наглядного выделения отдельных частей программы.

Программа, находящаяся в памяти компьютера, может быть выведена на экран с помощью команды LIST. Если программа не помещается на экране, то в нижней части экрана появляется запрос: scroll? Если в ответ нажать на клавишу BREAK, N, STOP или SPACE, то дальнейший вывод программы будет прекращен и появится сообщение:

D BREAK — CONT repeats, 0:1

Нажатие на любую другую клавишу приведет к высвечиванию следующей страницы программы.

Возможно также начать просмотр программы со строки с заданным номером с помощью команды LIST n, где n — номер строки, с которой будет осуществляться вывод программы на экран.

1.4. Запуск программы на Бейсике.

Для того, чтобы запустить программу на Бейсике, находящуюся в памяти компьютера, необходимо ввести команду RUN. Программа начнет выполняться с первого оператора до последнего, либо до появления оператора STOP. Запуск программы, начиная с заданной строки, производится с помощью команды RUN n, где n — номер строки, с которой должно начинаться выполнение программы.

Если в процессе выполнения программы интерпретатор обнаружит ошибку (например, переменной не присвоено значение), то выполнение программы будет прервано, и в двух нижних строках экрана будет выведено сообщение об ошибке (сообщения системы приведены в приложении 2). В случае успешного завершения программы в нижней части экрана выводится сообщение:

o OK, n : k

где α — номер сообщения,

OK — содержание сообщения (все правильно),

n — номер последней выполненной строки,

k — номер оператора в строке.

В случае выполнения оператора STOP выводится сообщение:

9 STOP statement, $n : k$

Выполнение программы после оператора STOP можно продолжить, выполнив в непосредственном режиме команду CONTINUE.

Описанные возможности интерпретатора можно использовать для тестирования и отладки программ, вставив дополнительные операторы STOP в критические точки программы.

Программу на Бейсике можно также запустить с помощью команд GO TO, GO TO n (а подпрограмму — GO SUB). Однако, команда GO TO не эквивалентна команде RUN. Команда RUN, кроме запуска программы, реализует также дополнительные функции: стирает область переменных программы и устанавливает значения некоторых системных переменных.

Прерывание выполнения программы на Бейсике может быть выполнено одновременным нажатием клавиш CAPS SHIFT и BREAK. Возобновить выполнение программы, как и в случае останова по оператору STOP, можно введя команду CONTINUE. Если программа была прервана во время операции ввода-вывода (сообщение: D BREAK — CONTINUE repeats, $n : k$), то после ввода команды CONTINUE выполнение прерванного оператора повторится с начала. В остальных случаях при прерывании программы выводится сообщение:

L BREAK into program, $n : k$

После ввода команды CONTINUE интерпретатор продолжает выполнение программы со следующего оператора.

Для прерывания выполнения оператора INPUT необходимо выполнить:

- в случае операторов INPUT a , либо INPUT $a\$$ — перевести курсор в начало строки командой "←" (CS и 5) и выполнить команду STOP;
- в случае оператора INPUT LINE $a\$$ прерывание осуществляется одновременным нажатием на CS и 6.

В обоих случаях выводится сообщение:

H STOP in INPUT, $n : k$

1.5. Загрузка и выгрузка Бейсик-программы.

В качестве внешней памяти для хранения программ пользователя в

компьютерах класса Спектрум наиболее часто используется бытовой кассетный магнитофон. Для того, чтобы выгрузить на ленту занесенную в память компьютера программу, нужно выполнить команду SAVE "имя". Имя файла является обязательным элементом команды, иначе появляется сообщение:

F Invalid file name, n : k

Для автоматического запуска программы после ввода она должна быть выведена с помощью команды:

SAVE "имя" LINE n

где n — номер строки в программе, с которого она будет стартовать после ввода.

Ввод программы с ленты производится по команде:

LOAD "имя"

Имя команды может быть пустым. В этом случае вводится первая встретившаяся программа. В отличие от имен переменных в Бейсике Спектрума, имена файла, написанные прописными и строчными буквами, являются различными именами.

Для ввода программы с магнитной ленты может быть также использована команда:

MERGE "имя"

В отличие от команды LOAD эта команда не стирает память перед вводом, а добавляет новые строки к ранее введенной программе. Строки с совпадающими номерами заменяются на вновь введенные.

2. ОБЩИЕ СВЕДЕНИЯ О ПРОГРАММИРОВАНИИ НА БЕЙСИКЕ.

2.1. Структура программы.

Программа на Бейсике представляет собой последовательность строк, имеющих следующий формат:

`<нс> оператор Бейсика [:оператор Бейсика...]`

где `<нс>` — номер строки. Номер строки должен быть целым числом и лежать в диапазоне $1 \div 9999$.

Операторы Бейсика делятся на два класса: выполняемые и невыполняемые. Выполняемые операторы определяют конкретные действия. Невыполняемые операторы описывают характеристики и значения данных, определяемые пользователем функции и т.д. Если в строке находится более одного оператора, они должны быть разделены двоеточием.

В программе могут быть помещены комментарии, начинающиеся с ключевого слова REM.

2.2. Алфавит.

Подобно любому языку Бейсик имеет алфавит, определяющий набор допустимых символов, из которых образуются элементы программы. В Бейсике Спектрума он состоит из букв, цифр, ключевых слов и специальных знаков. К буквенным символам относятся буквы латинского алфавита. Для интерпретатора безразлично, вводится прописная или строчная буква. К цифровым — цифры от 0 до 9. Ключевые слова Бейсика вводятся в Спектруме нажатием одной соответствующей клавиши. Ключевые слова Бейсика, введенные побуквенно, могут быть использованы в качестве имен переменных. Полный перечень ключевых слов Бейсика, используемых в Бейсике Спектрума, приведен в табл. 2. 1.

Таблица 2.1

Ключевое слово	Клавиша	Режим	Значение
ABC	G	E	Функция ABC
ACS	W + SS	E	Функция ACS
AND	Y + SS	K, L, C	Операция логического И (компоновция)
ASN	Q + SS	E	Функция ASN
AT	I + SS	K, L, C	Функция AT
ATN	E + SS	E	Функция ATN
ATTR	L + SS	E	Функция ATTR
BEEP	Z + SS	E	Оператор BEEP
BIN	B	E	Функция BIN
BORDER	B	K	Оператор BORDER
BRIGHT	B + SS	E	Оператор BRIGHT
CAT	9 + SS	E	Команда CAT
CHR\$	U	E	Функция CHR\$
CIRCLE	H + SS	E	Оператор CIRCLE
CLEAR	X	K	Оператор CLEAR
CLOSE#	5 + SS	E	Оператор CLOSE#
CLS	V	K	Оператор CLS
CODE	I	E	Функция CODE
CONTINUE	C	K	Команда CONTINUE
COPY	Z	K	Команда COPY
COS	W	E	Функция COS
DATA	D	E	Оператор DATA
DEF FN	I + SS	E	Оператор DEF FN
DIM	D	K	Оператор DIM
DRAW	W	K	Оператор DRAW
ERASE	7 + SS	E	Оператор ERASE
EXP	X	E	Функция EXP
FLASH	V + SS	E	Оператор FLASH
FN	2 + SS	E	Функция FN
FOR	F	K	Оператор FOR ... NEXT
FORMAT	0 + SS	E	Оператор FORMAT
GOSUB	H	K	Оператор GOSUB
GOTO	G	K	Оператор GOTO
IF	U	K	Оператор IF ... THEN
IN	I + SS	E	Функция IN
INK	X + SS	E	Оператор INK
INKEY\$	N	E	Функция INKEY\$
INPUT	I	K	Оператор INPUT

Ключевое слово	Клавиша	Режим	Значение
INT	R	E	Функция INT
INVERSE	M + SS	E	Оператор INVERSE
LEN	K	E	Функция LEN
LET	L	K	Оператор LET
LINE	3 + SS	E	Часть операторов INPUT и SAVE
LIST	K	K	Команда LIST
LLIST	V	E	Команда LLIST
LN	Z	E	Функция LN
LOAD	J	K	Команда LOAD
LPRINT	C	E	Команда LPRINT
MERGE	T + SS	E	Команда MERGE
MOVE	6 + SS	E	Оператор MOVE
NEW	A	K	Команда NEW
NEXT	N	K	Оператор FOR ... NEXT
NOT	S + SS	K, L, C	Операция логического отрицания
OPEN#	4 + SS	E	Оператор OPEN#
OR	U + SS	K, L, C	Операция логическ. ИЛИ (дизъюнкция)
OUT	0 + SS	E	Оператор OUT
OVER	N + SS	E	Оператор OVER
PAPER	C + SS	E	Оператор PAPER
PAUSE	M	K	Оператор PAUSE
PEEK	O	E	Функция PEEK
PI	M	E	Функция PI
PLOT	Q	K	Оператор PLOT
POINT	8 + SS	E	Функция POINT
POKE	O	K	Оператор POKE
PRINT	P	K	Оператор PRINT
RANDOMIZE	T	K	Оператор RANDOMIZE
READ	A	E	Оператор READ
REM	E	K	Оператор REM
RESTORE	S	E	Оператор RESTORE
RETURN	Y	K	Оператор RETURN
RND	T	E	Функция RND
RUN	R	K	Команда RUN
SAVE	S	K	Команда SAVE
SCREEN\$	K + SS	E	Функция SCREEN\$
SGN	F	E	Функция SGN
SIN	Q	E	Функция SIN
SQR	H	E	Функция SQR

Ключевое слово	Клавиша	Режим	Значение
STEP	D + SS	K, L, C	Часть оператора FOR ... NEXT
STOP	A + SS	K, L, C	Оператор STOP
STR\$	Y	E	Функция STR\$
TAB	P	E	Функция табуляции
TAN	E	E	Функция TAN
THEN	G + SS	K, L, C	Часть оператора IF ... THEN
TO	F + SS	K, L, C	Часть оператора FOR ... NEXT, операция сечения строки
URS	L	E	Функция URS
VAL	J	E	Функция VAL
VAL\$	J + SS	E	Функция VAL\$
VERIFY	R + SS	E	Команда VERIFY

Специальные знаки приведены в табл. 2.2.

Таблица 2.2

Знак	Клавиша	Режим	Значение
.	M + SS	K, L, C	Десятичная точка.
,	N + SS	K, L, C	Разделитель, вызывающий табуляцию
;	O + SS	K, L, C	Разделитель, подавляющий перенос строки.
'	7 + SS	K, L, C	Апостроф, разделитель, вызывающий перенос строки.
:	Z + SS	K, L, C	Разделитель операторов в строке.
(	8 + SS	K, L, C	Круглая скобка левая.
)	9 + SS	K, L, C	Круглая скобка правая.
+	K + SS	K, L, C	Плюс или знак сцепления строк.
-	J + SS	K, L, C	Минус.
*	B + SS	K, L, C	Знак умножения.
/	V + SS	K, L, C	Знак деления.
↑	H + SS	K, L, C	Возведение в степень.
<	R + SS	K, L, C	Меньше.
>	T + SS	K, L, C	Больше.
<=	Q + SS	K, L, C	Меньше, равно.
>=	E + SS	K, L, C	Больше, равно.
<>	W + SS	K, L, C	Не равно.
"	P + SS	K, L, C	Кавычки, выделение строковых констант.

Для записи комментариев и строк символов могут использоваться все символы из набора символов Спектрума.

2.3. Константы.

Под константой понимается заданное в явном виде значение, которое используется в программе на Бейсике. Существует два типа констант: строковые и числовые. Строковые константы иногда будут называться строками символов, или просто строками.

Строковая константа представляет собой заключенную в кавычки последовательность символов. Строковые константы обычно используются для задания текстовой информации и могут быть произвольной длины. Ниже приводятся примеры строковых констант:

"ZX SPECTRUM"

"88.000.000"

Числовые константы представляют собой положительные или отрицательные числа. Для положительных чисел задание знака "плюс" не обязательно. Константы могут быть записаны в форме целых чисел, чисел с плавающей точкой, чисел в экспоненциальной форме. Ниже приведены примеры констант:

12345678

0.935

15E3

0.1e-6

При выводе интерпритатор сам определяет форму представления чисел, так вышеперечисленные константы будут выведены компьютером в виде:

12345678

0.935

15000

1E-7

Так как для запоминания каждого числа в Спектруме отводится пять байтов памяти, то числа представимы с ограниченной точностью — 8 значащих цифр. Наибольшее число, представленное в Спектруме: 1.7014118E+38. Наименьшее: -1.7014118E+38. Наименьшее положительное число: 1E-38.

2.4. Переменные.

Переменные — это именованный объект, который может принимать различные значения в процессе выполнения программы. В Бейсике существуют два типа переменных: числовые переменные и переменные типа строка символов или строковые переменные.

Значениями строковой переменной могут быть строки символов. Длина строковой переменной не фиксирована.

Переменная получает значение в результате присваивания ей значения другой переменной, константы, результата вычислений или посредством оператора ввода-вывода. В любом случае это значение должно соответствовать типу переменной, которой оно присваивается.

Имена строковых переменных должны состоять из одной буквы и символа \$, например, A\$, B\$, R\$.

Имена числовых переменных должны начинаться с буквы, могут иметь произвольную длину и состоять из букв и цифр. Пробелы, вставленные в имя переменной, не изменяют его. Ниже приведены примеры допустимых имен переменных:

A 567	} это одно и то же имя
a567	
X	
Variable	

2.5. Массивы.

Массив — это упорядоченная последовательность однотипных величин, обращение к которым может осуществиться при помощи одного и того же имени (имени массива). Каждое отдельное значение в массиве называется элементом массива. Обращение к элементу массива осуществляется при помощи имени массива, снабженного индексами. Элемент массива является переменной и может быть использован в любом месте, где может быть использована переменная.

Задание имени массива, типа и количества его элементов, а также резервирование памяти под массив называется объявлением массива и выполняется с помощью оператора DIM. Во время объявления массива строковым переменным присваивается значение, равное пустой строке, а числовым переменным — нуль. Именем массива должна быть одна буква для числовых массивов и одна буква с символом \$ для строковых массивов. Пример объявления массивов:

10 DIM A(3,3)

20 DIM B\$(10,7)

Оператором 10 объявляется двумерный числовой массив из 9 элементов, а оператором 20 — строковый массив из 10 строк по семь символов в строке.

Для обращения к элементу массива используется имя массива с индексами. Количество индексов равно количеству измерений массива. Минимальным значением индекса массива является значение 1. Количество измерений массива и количество элементов в одном измерении ограничено только объемом доступной памяти. При обращении к элементу числового массива необходимо указывать столько индексов, сколько размерностей имеет соответствующий массив. При обращении к элементу строкового массива обращение может производиться либо к целой строке (при этом указывается количество индексов без последнего, определяющего символа в строке), либо до единичного символа (при этом последний индекс должен быть указан).

2.6. Выражения.

Выражение представляет собой комбинацию переменных, констант и обращений к функциям, соединенных знаками операций. Порядок выполнения операций при вычислении выражения может регулироваться с помощью скобок. В простейшем случае выражение может состоять из одной переменной или константы. В Бейсике допустимы выражения двух типов: числовые и строковые. В числовых выражениях могут быть использованы арифметические операции, операции сравнения и логические операции. Арифметические операции выполняют действия обычной арифметики. В табл.2.3 приведены все арифметические операции Спектрума в порядке их приоритетов.

Таблица 2.3

Знак операции	Название операции	Пример
$\uparrow$	Возведение в степень	$A \wedge 2$
$*, /$	Умножение и деление	$2 * 2, 6 / \cos \alpha$
$+, -$	Сложение и вычитание	$A + B, C - D$

Операции сравнения выполняют сравнение двух значений. Сравниваемые значения должны быть одинакового типа: либо оба числовые, либо оба строковые. Результатом сравнения является числовое значение 1, если между этими значениями выполняется заданное отношение, выраженное знаком операции, или значение 0 в противном случае. Этот результат может быть использован для принятия решения относительно дальнейшего хода выполнения программы, например, в операторе IF. Операторы сравнения перечислены в табл. 2.4.

Таблица 2.4

Знак операции	Название операции	Пример
=	Равно	$A = B$
< >	Не равно	"a" < > "b"
<	Меньше	$3 < 5$
< =	Меньше или равно	$C < = B$
>	Больше	$A\$ > B\$$
> =	Больше или равно	$C > = d$

Все операции сравнения имеют в Спектруме одинаковый приоритет и выполняются слева направо. При объединении в одном выражении арифметических операций и операций сравнения первыми выполняются арифметические операции. Так результат вычисления выражения $6 < 5 + 4$ будет равен 1, так как сложение выполняется перед сравнением. Этот порядок может быть изменен с помощью скобок.

Бейсик позволяет использовать в выражениях операции сравнения строк символов. Такое сравнение выполняется посимвольно слева направо. Если все символы совпадают, строки считаются равными. Если встретилась пара неравных символов, то большей считается строка, содержащая символ с большим кодом в таблице кодов. Если сравниваются строки разной длины, то более короткая строка считается меньшей.

Логические операции, используемые в Бейсике Спектрума, приведены в табл.2.5 в порядке их приоритетов.

Таблица 2.5

Знак операции	Название операции	Пример
NOT	Логическое отрицание	NOT A
AND	Логическая конъюнкция	B AND C
OR	Логическая дизъюнкция	5 OR 6

Логические операции выполняются только над числовыми значениями. Результатом логической операции является:

NOT A = 1, если A = 0

NOT A = 0, если A $\neq$ 0

A AND B = 1, если A $\neq$ 0 и B $\neq$ 0

A AND B = 0, если либо A = 0, либо B = 0

$A \text{ OR } B = 1$, если либо $A \neq 0$, либо $B \neq 0$

$A \text{ OR } B = 0$, если $A = 0$ и $B = 0$.

2.7. Порядок выполнения операций.

Вычисление значения выражения выполняется в следующем порядке:

- вычисляются значения функций;
- выполняются арифметические операции с учетом приоритетов;
- выполняются операции сравнения;
- выполняются логические операции в соответствии с приоритетом.

Операции одного уровня выполняются слева направо. Для изменения порядка выполнения операций могут использоваться скобки. Операции в скобках выполняются первыми. Внутри скобок используется обычный порядок выполнения операций.

2.8. Выражения над строками символов.

В выражениях над строками символов (строковых выражениях) могут участвовать только данные типа строка символов. Результатом вычисления строкового выражения является строка символов. В строковых выражениях допустимы только две операции:

- операция сцепления строк;
- операция сечения строки.

Над строками символов можно выполнять также операции сравнения, однако эти операции не могут использоваться в строковых выражениях, так как результат их выполнения является числовым значением.

Операция сцепления обозначается знаком плюс (+). В результате этой операции несколько строк символов соединяются в одну.

Например:

```
10 LET A$ = "Rostov"
```

```
20 LET B$ = "-on-"
```

```
30 LET C$ = "Don"
```

```
40 PRINT A$+B$+C$
```

```
RUN
```

```
Rostov-on-Don
```

Операция сечения служит для выделения подстрок из целой строки. Формат операции сечения:

<строковое выражение> ([выражение] TO [выражение])

Пример операции сечения: 10 LET A\$ = "ABCDEF" (2 TO 5)

```
20 PRINT A$
```


RUN
BCDE

Допустимы следующие сокращения формата операции сечения, например:

```
10 PRINT "ABCDEF" (TO 5)
RUN
ABCDE
```

```
10 PRINT "ABCDEF" (2 TO)
RUN
BCDEF
```

```
PRINT "ABCDEF" (TO)
ABCDEF
```

Если выражение, стоящее слева от ключевого слова TO, будет больше выражения, стоящего справа, то результатом операции будет пустая строка. Если выражение, стоящее справа, превышает по величине количество символов в строке, будет выдано сообщение:

3 Subscript wrong, n : k

Приоритет операции сечения выше, чем приоритет операции сцепления строк.

2.9. Функции.

Функции используются в выражениях для обращения к заранее определенным операциям. При вычислении выражения обращение к функции заменяется значением, возвращаемым функцией. Возвращаемые значения могут быть или числовыми или строковыми. Функция может использоваться в выражении только в том случае, если тип возвращаемого ею значения совпадает с типом выражения.

Бейсик Спектрума включает набор встроенных функций, которые реализуют часто встречаемые операции. Все встроенные функции описаны в разд.3.

Кроме того, Бейсик допускает использование функций, определяемых пользователем. Такие функции используются для реализации алгоритмов, часто встречающихся в конкретной программе. Для определения таких функций служит оператор DEF FN.

2.10. Работа с экраном.

Бейсик Спектрума позволяет выводить на экран текст, символы псевдографики и графики, определяемой пользователем. С помощью специальных операторов можно нарисовать на экране точки, линии, дуги и окружности, а также различные композиции, состоящие из этих элементов. Изображение может быть окрашено в 8 цветов, включая белый и черный, и иметь два уровня яркости. Возможно также использование режима мерцания.

Изображение в Спектруме генерируется на основании содержания специальной области памяти компьютера, называемой экранной областью. Эта область начинается с адреса 16384. Каждой точке (пикселю) на экране соответствует один бит в экранной области памяти. Если бит в экранной области равен 1, то соответствующая ему точка на экране имеет цвет «чернил». Если бит равен 0, то точка имеет цвет фона.

Изображение на экране состоит из 192 строк по 256 точек в строке. То есть экран является матрицей из 192×256 элементов, для отображения которых требуется 49152 бита или 6144 байта памяти. Связь координат точки на экране с адресом в экранной области определяется довольно сложным выражением:

адрес $(X,Y) = 16384 + 32 * (\text{INT}((175 - Y)/8) - \text{INT}((175 - Y)/64) * 8 + 8 * (175 - Y - \text{INT}((175 - Y)/8) * 8 + 64 * \text{INT}((175 - Y)/64)) + \text{INT}(X/8)$.
При этом должен быть установлен бит с номером: $7 - X + \text{INT}(X/7) * 7$.

Продемонстрировать эту зависимость можно последовательно загружая данные в экранную область Спектрума, например, по следующей программе:

```
10 FOR i=16384 TO 22527
20 POKE i, 255
30 NEXT i
40 PAUSE 0
RUN
```

В процессе выполнения программы видно, что экран разделен на три части, каждая из которых состоит из 64 строк по 256 точек. В начале загрузки высвечиваются точки в 0, 8, 16, ..., 56-й строке верхней части экрана. Далее высвечиваются точки в строках 1, 9, 17, ..., 57 и т.д. Эта процедура повторяется до момента высвечивания всех точек в верхней трети экрана. Аналогично высвечиваются точки в центральной и нижней части экрана.

Изображение на экране в Спектруме может быть раскрашено. Для этого нужно сообщить компьютеру дополнительную информацию: цвет, яркость,

мерцание. Однако, если бы эта информация задавалась для каждой точки изображения, то для ее запоминания потребовалась бы большая часть памяти компьютера. Поэтому в Спектруме эта информация, называемая атрибутами, задается для целой группы точек, образующих матрицу 8*8, или знакоместо.

Атрибуты знакоместа запоминаются в специальной области памяти, называемой областью атрибутов, начиная с адреса 22528. Каждое знакоместо подчинено одному байту из области атрибутов. Формат байта области атрибутов приведен на рис.3.

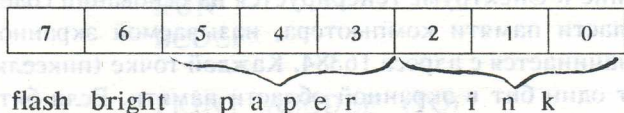


Рис.3. Формат байта атрибутов

Биты 0 ÷ 2 определяют цвет "чернил" в знакоместе.

Биты 3 ÷ 5 определяют цвет фона.

В Спектруме принята следующая кодировка цвета:

- 0 — черный
- 1 — синий
- 2 — красный
- 3 — пурпурный
- 4 — зеленый
- 5 — голубой
- 6 — желтый
- 7 — белый

Бит 6 определяет яркость: 1 — повышенная, 0 — нормальная.

Бит 7 определяет мерцание: 1 — включено, 0 — выключено.

Бейсик Спектрума позволяет управлять также цветом граничной части экрана (бордюром). Цвет бордюра задается при помощи специального оператора BORDER n, где n — число, определяющее цвет согласно принятой кодировке.

Операционная система Спектрума позволяет выводить на экран не только отдельные точки, но и символы. Символ занимает одно знакоместо 8*8 точек. Таким образом, экран в символьном режиме может быть представлен как 24 строки по 32 знакоместа. Строки в Спектруме нумеруются от 0 до 23, а столбцы от 0 до 31. Нумерация строк и столбцов начинается в левом верхнем углу экрана.

Верхние 22 строки экрана предназначены для:

- распечаток пользователя;
- распечаток текста программ - листингов программ.

Две нижние строки, так называемое системное окно, используются для:

- ввода и редактирования с помощью клавиатуры строк программы;
- ввода данных с помощью оператора INPUT;
- для сообщений системы.

Если одна строка программы занимает места больше, чем две нижние строки экрана, то системное окно расширяется вверх на необходимое количество строк. Если вводимая строка занимает места больше, чем весь экран, то подается звуковой сигнал и вновь вводимые символы не высвечиваются на экране, однако, вводятся в память.

3. ОПЕРАТОРЫ, ФУНКЦИИ И КОМАНДЫ.

3.1. Общие сведения.

В этом разделе в алфавитном порядке описываются операторы, функции и команды Бейсика. Описание выполнено по схеме: назначение, формат, характеристика операндов, правила применения и особенности выполнения, примеры.

Разделение на команды и операторы традиционно для интерпретатора. Команды, так как они оперируют целиком программами, чаще используются в режиме прямого выполнения. Операторы служат для реализации алгоритма программы. Большинство команд и операторов могут использоваться как в режиме прямого выполнения, так и в составе программы.

Встроенные функции являются составной частью интерпретатора Бейсика. Они могут использоваться в программе в любом месте, где могут быть использованы переменные. Исключение составляет использование их сле-
ва от знака присваивания.

3.2. Функция ABS.

НАЗНАЧЕНИЕ. Вычисление абсолютной величины заданного числа.
ФОРМАТ.

ABS X

где X — число, переменная, либо выражение в скобках.

ПРИМЕР.

PRINT ABS (7*(-8))

56

3.3. Функция ACS.

НАЗНАЧЕНИЕ. Вычисление арккосинуса заданного числа.

ФОРМАТ.

ACS X

где X — число, переменная, либо выражение в скобках.

ОПИСАНИЕ. Результатом выполнения функции ACS является значение угла, синус которого равен значению выражения X. Значение выражения X должно лежать в диапазоне от минус 1 до 1. В противном случае выдается сообщение:

A Invalid argument, n : k

Угол представлен в радианах и может принимать значение от 0 до π , где $\pi = 3,1415927$. Для преобразования угла из радианной меры в градусную необходимо значение угла умножить на число $180/\pi$

ПРИМЕР.

100 LET R = ACS0.5

110 LET g = R*180/PI

120 PRINT R, g

RUN

1.0471976

60

В этом примере с помощью функции ACS определяется угол, косинус которого равен 0,5.

3.4. Функция ASN.

НАЗНАЧЕНИЕ. Вычисление арксинуса заданного числа.

ФОРМАТ.

ASN X

где X — число, переменная, либо выражение в скобках.

ОПИСАНИЕ. Результатом выполнения функции ASN является значение угла, синус которого равен значению выражения X. Значение выражения X должно лежать в диапазоне от минус 1 до 1. Угол представлен в радианах и может принимать значение от $-\pi/2$ до $\pi/2$.

ПРИМЕР.

```
PRINT ASN(RND*2 - 1)*2+PI
```

В этом примере с помощью функций ASN и RND выводится случайное значение угла в диапазоне от 0 до 2π .

3.5. Функция ATN.

НАЗНАЧЕНИЕ. Вычисление арктангенса заданного числа.
ФОРМАТ.

```
ATN X
```

где X — число, переменная, либо выражение в скобках.

ОПИСАНИЕ. Результатом выполнения функции ATN является значение угла, тангенс которого равен значению выражения X. Угол представлен в радианах и может принимать значения в диапазоне от минус $\pi/2$ до $\pi/2$.

ПРИМЕР.

```
PRINT ATN1*180/PI
```

```
45
```

3.6. Функция ATTR.

НАЗНАЧЕНИЕ. Дает число, показывающее состояние цветовых атрибутов в заданном знакоместе экрана.

ФОРМАТ.

```
ATTR(W,K)
```

где W, K — любые числовые выражения, значения которых являются координатами точки на экране. Значение W — номер строки, K — номер столбца. Значение W должно лежать в пределах от 0 до 23, а значение K от 0 до 31. Содержание байта поля области атрибутов можно определить уравнением:

```
ATTR = 128*FLASH + 64*BRIGHT + 8*PAPER + INK.
```

Адрес байта в поле области атрибутов с координатами W, K можно определить по формуле:

```
22528 + 32*W + K.
```

ПРИМЕР.

```
NEW
```

```
PRINT ATTR(0,1)
```

```
56
```


3.7. Оператор BEEP.

НАЗНАЧЕНИЕ. Подача звукового сигнала.

ФОРМАТ.

BEEP t,f

t — длительность в секундах (от 0 до 10,499),

f — высота в полутонах (от -60 до +69 относительно "до" первой октавы); t и f могут быть любыми цифровыми выражениями.

ПРИМЕР.

```
10 FOR I = -60 TO 69
```

```
20 BEEP 0.5,I
```

```
30 NEXT I
```

В этом примере генерируются все тона длительностью 0,5 сек.

3.8. Функция BIN.

НАЗНАЧЕНИЕ. Перевод числа из двоичной формы в десятичную.

ФОРМАТ.

BIN B

где B — двоичное число, содержащее не более 16 цифр.

ПРИМЕР.

```
PRINT BIN 10101010101010
```

```
43690
```

3.9. Оператор BORDER.

НАЗНАЧЕНИЕ. Установка цвета бордюра.

ФОРМАТ.

BORDER N

где N — число, переменная, либо выражение.

ОПИСАНИЕ. При выполнении оператора BORDER значение N округляется до ближайшего целого и определяет цвет бордюра. Значение должно лежать в диапазоне от 0 до 7.

ПРИМЕР.

```
BORDER 3
```

В этом примере устанавливается пурпурный цвет бордюра.

3.10. Оператор BRIGHT.

НАЗНАЧЕНИЕ. Установка яркости.

ФОРМАТ.

BRIGHT i

где i - число, переменная, либо выражение. Значением i должны быть числа 0, 1, 8.

ОПИСАНИЕ. Оператор BRIGHT определяет уровень яркости изображения в знакоместе при выводе очередных символов на экран. Значение 0 аргумента BRIGHT задают нормальную яркость изображения, значение 1 — повышенную. Значение 8 задаст использование значения атрибута BRIGHT, взятого из байта памяти области атрибутов. Уровень яркости может быть задан глобально (для всех последующих вводимых символов) или локально, если ключевое слово BRIGHT входит в состав операторов PRINT, INPUT, PLOT и т.д. В последнем случае действие заданного режима распространяется только в пределах действия одного оператора.

ПРИМЕР.

```
10 PAPER 1: INK 7
20 CLS
30 PRINT "GLOBLE"
40 PRINT PAPER 6; INK 2; BRIGHT 1; FLASH 1; "LOCAL"
50 PAPER 2
60 PRINT "GLOBLE"
RUN
```

Оператор 10 устанавливает синий цвет фона и белый цвет чернил, цвет экрана не меняется. Оператор 20 стирает экран, цвет фона становится синим. Оператор 30 выводит слово GLOBLE в первой строке белыми буквами на синем фоне. Оператор 40 выводит во второй строке слово LOCAL красными буквами на желтом фоне, включены мерцание и повышенная яркость. Оператор 50 устанавливает красный цвет фона. Оператор 60 выводит слово GLOBLE белыми буквами на красном фоне.

3.11. Команда CAT.

НАЗНАЧЕНИЕ. Распечатка каталога картриджа микродрайва.

ФОРМАТ.

CAT

3.12. Функция CHR\$.

НАЗНАЧЕНИЕ. Получение символа, соответствующего заданному коду в таблице кодов ЭВМ.

ФОРМАТ.

CHR\$ N

где N — число, переменная, либо выражение в скобках.

ОПИСАНИЕ. Результатом выполнения функции CHR\$ является символ, код которого в таблице кодов компьютера равен значению N. Значение N, округленное до ближайшего целого, должно лежать в диапазоне от 0 до 255.

П Р И М Е Р .

```
10 CLS
```

```
20 FOR i = 32 TO 255
```

```
30 PRINT i, CHR$ i
```

```
40 NEXT i
```

```
RUN
```

Приведенная в примере программа выводит на экран все символы, кроме управляющих, и соответствующие им коды.

3.13. Оператор CIRCLE.

НАЗНАЧЕНИЕ. Вывод окружности на экран дисплея.

ФОРМАТ.

CIRCLE [оператор задания атрибутов;...] X, Y, R
оператор задания атрибутов — операторы INK, PAPER,
FLASH, BRIGHT, OVER и (или) INVERSE;

X — абсцисса центра окружности;

Y — ордината центра окружности;

R — радиус окружности.

ОПИСАНИЕ. Оператор вычерчивает окружность радиусом R с центром в точке X, Y. X, Y, R могут быть любыми числовыми выражениями, но при этом должны выполняться неравенства:

$$X + R < 255$$

$$X - R \geq 0$$

$$Y + R \leq 255$$

$$Y - R \geq 0$$

Если какое-либо из неравенств не будет выполнено, то выполнение оператора прервется, а на экране высветится сообщение об ошибке:

В Integer out of range, n : k

П Р И М Е Р .

```
10 FOR r = 0 TO 81 STEP 9
```

```
20 CIRCLE 128, 88, r
```

30 NEXT r
RUN

Эта программа рисует девять окружностей с общим центром.

3.14. Оператор CLEAR.

НАЗНАЧЕНИЕ. Инициализация области данных программы.

ФОРМАТ.

CLEAR [a]

где a — любое числовое выражение

ОПИСАНИЕ. Оператор CLEAR очищает все области переменных, очищается область экрана (подобно CLS), выполняет функцию RESTORE, переустанавливает позицию PLOT в левый нижний угол экрана, устанавливает новую границу памяти, используемой Бейсиком, по адресу, равному значению a. Начальное значение a — 65376.

3.15. Оператор CLOSE#.

НАЗНАЧЕНИЕ. Закрыть поток.

ФОРМАТ.

CLOSE# n

где n — любое числовое выражение, значение которого должно лежать в диапазоне от 0 до 15.

ОПИСАНИЕ. Закрывает поток, открытый оператором OPEN#. Не могут быть закрыты ни один из стандартных каналов:

"K" — клавиатура;

"S" — экран;

"P" — принтер.

Закрытие такого канала приводит к его автоматическому открытию для потока данных со стандартным номером. Команда RUN закрывает все каналы кроме стандартных.

ВНИМАНИЕ! Попытка закрыть неоткрытые потоки с номерами 4, 5...9 приводит к "зависанию" и требует перезагрузки системы.

3.16 Оператор CLS.

НАЗНАЧЕНИЕ. Очистка экрана.

ФОРМАТ.

CLS

ОПИСАНИЕ. Очищает экран. Устанавливает атрибуты экранной области в соответствии с заданными операторами **INK**, **PAPER**, **FLASH**, **BRIGHT**. Пример см. п.3.10.

3.17. Функция **CODE**.

НАЗНАЧЕНИЕ. Дает код первого символа строки.

ФОРМАТ.

CODE A\$

где **A\$** — строка, строковая переменная.

ОПИСАНИЕ. Допустимо использование в операторе **CODE** сечений строк. Сцепления строк в операторе **CODE** должны быть заключены в скобки.

ПРИМЕР.

PRINT CODE ("abc"+"def")(1 TO 2)

97

3.18. Команда **CONTINUE**.

НАЗНАЧЕНИЕ. Продолжение исполнения программы после остановки.

ФОРМАТ.

CONTINUE

ОПИСАНИЕ. Команда **CONTINUE** может быть использована для возобновления выполнения программы после прерывания, вызванного нажатием **CS** и **BREAK**, или выполнения оператора **STOP**.

3.19. Команда **COPY**.

НАЗНАЧЕНИЕ. Команда **ZX** принтера. Выполняет копирование экрана на бумагу.

ФОРМАТ.

COPY

ОПИСАНИЕ. Выполняет на печатающем устройстве полную копию изображения на экране.

3.20. Функция COS.

НАЗНАЧЕНИЕ. Вычисление косинуса заданного угла.

ФОРМАТ.

COS X

где X — число, переменная, либо выражение в скобках.

ОПИСАНИЕ. Функция вычисляет косинус угла, значение которого в радианах равно X. Для преобразования значения угла из градусной меры в радианную нужно значение угла умножить на $\pi / 180$, где $\pi = 3,1415927$.

ПРИМЕР.

```
10 INPUT X
20 LET r = X*PI/180
30 PRINT X, COS r
40 GO TO 10
RUN
```

Данная программа вычисляет косинус любого введенного угла и печатает значение угла и его косинуса.

3.21. Оператор DATA.

НАЗНАЧЕНИЕ. Запоминание данных для оператора READ.

ФОРМАТ.

DATA C1[,C2,C3...]

где C1, C2... — числовые и строковые константы.

ОПИСАНИЕ. Оператор DATA является невыполняемым и может быть помещен в любом месте программы. В программе может использоваться любое количество операторов DATA. Данные, определенные в программе оператором DATA, рассматриваются как непрерывный список констант, расположенных в порядке возрастания номеров строк, содержащих операторы DATA.

Пример использования оператора DATA приведен при описании оператора READ (п.3.62).

3.22. Оператор DEF FN.

НАЗНАЧЕНИЕ. Задает функцию пользователя.

ФОРМАТ.

DEF FN A ([X, Y...])=Z

A — однобуквенное имя, используемое для вызова функции;
X, Y — однобуквенные имена параметров, которые получают значения во время вызова функции. Во время вызова функции параметры получают значения аргументов, которые заданы в вызове функции;

Z — выражение, которое определяет функцию. Тип выражения должен соответствовать типу, объявленному именем функции.

ОПИСАНИЕ. Аргументы, которые появляются в выражении определения, служат только для определения функции, они не активируют программные переменные, которые имеют то же имя. Тип функции определяет числовое или строковое значение, образуемое функцией. Если тип определения не соответствует типу функции, то появляется сообщение об ошибке. Если функция определена более одного раза, то действительным является самое первое выполненное определение.

ПРИМЕР:

```
10 DEF FN C(R)=2*PI*R
20 INPUT "radius? —"; x
30 PRINT "circle ="; FN C(x)
RUN
radius? — 1
circle = 6.2831853
```

В этом примере функция FN C(x) используется для вычисления длины окружности.

3.23. Оператор DIM.

НАЗНАЧЕНИЕ. Определение массива.

ФОРМАТ.

`DIM A(X [Y...])`

A — однобуквенное имя массива;

X, Y... — числовое выражение, определяющее границу измерения массива. Значение выражения, округленное до ближайшего целого, определяет границу измерения и должно быть положительным числом больше нуля.

ОПИСАНИЕ. Оператор DIM устанавливает исходные значения элементов массива в нуль, уничтожает все массивы с тем же именем. Для строковых массивов выражение для последней размерности определяет число символов в строке.

ПРИМЕР.

```
10 DIM(2,2)
20 INPUT "A(1,1)?"; A(1,1)
30 INPUT "A(1,2)?"; A(1,2)
40 INPUT "A(2,1)?"; A(2,1)
50 INPUT "A(2,2)?"; A(2,2)
60 LET d=A(1,1)*A(2,2)-A(1,2)*A(2,1)
70 PRINT "det ="; d
RUN
```

Программа, приведенная в данном примере, вычисляет определитель двумерной матрицы.

3.24. Оператор DRAW.

НАЗНАЧЕНИЕ. Вычерчивание прямых линий, дуг окружностей.
ФОРМАТ.

DRAW [оператор задания атрибутов;...] X, Y [,O] оператор задания атрибутов — операторы **INK**, **PAPER**, **FLASH**, **BRIGHT**, **OVER** и (или) **INVERSE**;

X — абсцисса;

Y — ордината;

O — угол в радианах.

X, Y, O могут быть любыми числовыми выражениями. Значения X, Y для любой точки нарисованной линии (или дуги) не должны выходить за пределы экрана:

$$0 \leq X \leq 255$$
$$0 \leq Y \leq 175$$

ОПИСАНИЕ. Вывод с помощью оператора **DRAW** начинается либо в той точке, где закончилось последнее выполнение операторов **PLOT**, **DRAW** или **CIRCLE**, либо в левом нижнем углу (после выполнения операторов **RUN**, **CLEAR**, **CLS**, **NEW**). Конечная точка — это точка, отстоящая на X графических точек вправо и на Y вверх от исходной. Оператор **DRAW** сам не определяет начальную точку. Числа X, Y, O могут быть отрицательными.

При наличии параметра O можно рисовать дуги, причем O — это число радиан, на которое производится поворот при построении линии. Если $O > 0$, то поворот происходит против часовой стрелки, если $O < 0$, то по часовой стрелке.

ПРИМЕР.

```
10 PLOT 56, 0: DRAW 20, 160
20 PLOT 74, 144: DRAW 64, 0, -1: DRAW 64, 0, 1
30 DRAW -9, -72
40 DRAW -64, 0, -1: DRAW -64, 0, 1
RUN
```

Программа данного примера рисует на экране флаг.

3.25. Оператор ERASE.

НАЗНАЧЕНИЕ. Работа с микродрайвом.

ФОРМАТ.

```
ERASE A$
```

A\$ — строковое выражение.

3.26. Функция EXP.

НАЗНАЧЕНИЕ. Вычисление показательной функции.

ФОРМАТ.

```
EXP X
```

где X — число, переменная, либо выражение в скобках.

ОПИСАНИЕ. Эта функция возвращает значение, равное e в степени x; e является основанием натурального логарифма и равно 2.7182818.

ПРИМЕР.

```
PRINT EXP 1
```

```
2.7182818
```

3.27. Оператор FLASH.

НАЗНАЧЕНИЕ. Установка мерцания.

ФОРМАТ.

```
FLASH i
```

где i — число, переменная, либо выражение в скобках. Значением i могут быть числа 0, 1, 8.

ОПИСАНИЕ. Оператор FLASH определяет режим мерцания изображения в знакоместе при выводе очередных символов на экран. Значение 0 аргумента FLASH задает отсутствие мерцания изо-

бражения, значение 1 включает мерцание. Значение 8 задает использование значения атрибута FLASH, взятого из атрибута памяти области атрибутов. Режим мерцания может быть задан глобально (для всех последующих операторов) или локально, если ключевое слово FLASH входит в состав операторов PRINT, INPUT, PLOT и т.д. В последнем случае действие заданного режима распространяется только в пределах действия одного оператора. Пример смотри п.3.10.

3.28. Функция FN.

НАЗНАЧЕНИЕ. Вызов функции пользователя.

ФОРМАТ.

FN A(X,Y...)

A — однобуквенное имя функции;

X, Y — параметры функции.

ОПИСАНИЕ. Вызывает функцию пользователя A, заданную оператором DEF FN A. Аргумент должен быть в скобках. Допустимо использование выражений или вычисления параметров функции. Если аргумента нет, скобки все равно должны быть. Пример смотри п.3.22.

3.29. Операторы FOR и NEXT.

НАЗНАЧЕНИЕ. Оператор цикла.

ФОРМАТ.

FOR i=X TO Y [STEP Z]

NEXT i

i — однобуквенная переменная, которая будет использоваться в качестве счетчика выполнения цикла;

X — числовое выражение, значение которого задает начальное значение счетчика цикла;

Y — числовое выражение, значение которого задает конечное значение счетчика цикла;

Z — числовое выражение, значение которого задает приращение счетчика цикла.

ОПИСАНИЕ. Строки программы, следующие за оператором FOR, выполняются до тех пор, пока не будет встречен оператор NEXT.

В результате выполнения оператора NEXT счетчик цикла увеличивается на Z, либо по умолчанию на 1, и полученное значение сравнивается с Y. Если значение счетчика оказывается большим, чем заданное конечное значение, то цикл завершается, и выполнение программы продолжается с оператора, следующего за оператором NEXT. В противном случае осуществляется еще одно выполнение операторов, входящих в цикл, начиная с оператора, следующего за оператором FOR.

Если приращение задано отрицательным числом, то счетчик уменьшается при каждом проходе цикла, а цикл выполняется до тех пор, пока счетчик не станет меньше, чем конечное значение.

Если условие цикла сразу не выполняется, то цикл пропускается. При $Z=0$ получается бесконечный цикл.

После завершения цикла управляющая переменная принимает конечное значение плюс шаг.

Циклы могут быть вложенными друг в друга. Если циклы вложены, то каждый цикл должен иметь уникальное значение счетчика цикла. Оператор NEXT, заканчивающий внутренний цикл, должен появляться раньше, чем оператор NEXT, заканчивающий внешний цикл.

П Р И М Е Р .

```
10 FOR M =0 TO 6
20 FOR N =0 TO M
30 PRINT M; " "; N; " ";
40 NEXT N
50 PRINT
60 NEXT M
```

Эта программа выводит числа домино.

3.30. Оператор FORMAT.

НАЗНАЧЕНИЕ. Работа с микродрайвом.
ФОРМАТ.

FORMAT A\$

где A\$ — строковое выражение.

3.31. Операторы GO SUB и RETURN.

НАЗНАЧЕНИЕ. Переход к подпрограмме и возврат в нее.

ФОРМАТ.

GO SUB N

RETURN

N — любое числовое выражение, значение которого является номером начальной строки подпрограммы.

ОПИСАНИЕ. Оператор GO SUB осуществляет передачу управления заданной подпрограмме, выполнение которой должно завершаться оператором RETURN. По оператору RETURN осуществляется возврат к оператору, следующему за оператором GO SUB.

Подпрограмма может содержать более одного оператора RETURN, если необходимо осуществлять возврат из нескольких точек подпрограммы. Подпрограммы могут располагаться в любом месте программы.

Программа может вызываться из нескольких точек программы или из другой подпрограммы. Вложенность вызовов подпрограммы ограничена только объемом свободной памяти.

П Р И М Е Р .

```
10 PRINT "PROGRAM"
20 GO SUB 100
30 PRINT "PROGRAM"
40 STOP
100 PRINT "SUBROUTINE"
110 RETURN
RUN
PROGRAM
SUBROUTINE
PROGRAM
```

3.32. Оператор GO TO.

НАЗНАЧЕНИЕ. Переход к заданной строке программы.

ФОРМАТ.

GO TO N

N — любое числовое выражение.

ОПИСАНИЕ. Оператор GO TO прерывает нормальную последовательность выполнения операторов и передает управление строке, номер которой указан в операторе GO TO. Если значение N указывает на выполняемый оператор, то выполнение продолжается с этого оператора. Если операнд N указывает невыполняемый опе-

ратор (REM, DATA), то выполнение программы продолжается с первого выполняемого оператора, находящегося после указанной строки. Оператор GO TO может использоваться в непосредственном режиме для возобновления выполнения программы с любой точки. Эта возможность может оказаться полезной при отладке программы.

П Р И М Е Р .

```
10 INPUT "temperature deg C?"; C
20 LET f=C*9/5+32
30 PRINT C; "deg C ="; f; "deg F"
40 GO TO 10
```

В этой программе значения температуры в градусах Цельсия переводятся в температуру по Фаренгейту.

3.33. Оператор IF.

НАЗНАЧЕНИЕ. Управление ходом выполнения программы.
ФОРМАТ.

IF X THEN op [:op...]

X — любое числовое выражение;

op :op... — последовательность операторов Бейсика, разделенных двоеточием.

ОПИСАНИЕ. Если значение выражения X не равно нулю (истинно для операций сравнения и логических операций), то выполняется последовательность операторов после THEN. В противном случае выполняется переход к следующей строке. (Не к следующему оператору в этой строке, а именно к следующей строке! Типичная ошибка!)

П Р И М Е Р . Известно, что любую целочисленную денежную сумму $S > 7$ руб. можно выплатить без сдачи трешками и пятерками. Написать программу вычисления по заданному целому $S > 7$ всех пар целых неотрицательных чисел a и b, таких, что $S = 3a + 5b$. Для нахождения первой пары чисел a и b использовать тот факт, что в зависимости от значения остатка P и частного A от деления S на 3, сумма S может быть выражена либо только a трешками, либо двумя пятерками и a-3 трешками, либо a-1 трешкой и одной пятеркой.

НАПРИМЕР.

```
10 INPUT "S = ?"; S
20 LET a = INT(S/3): LET P = S-a*3: LET b=0
30 IF P=0 THEN GO TO 60
40 IF P=1 THEN LET a=a-3: LET b=2: GO TO 60
50 LET a=a-1: LET b=1
60 PRINT "S=3*"; a; "+5*"; b
70 IF a >= 5 THEN LET a=a-5: LET b=b+3: GO TO 60
RUN
S = ? 51
S = 3*17+5*0
S = 3*12+5*3
S = 3*7+5*6
S = 3*2+5*9
```

3.34. Функция IN.

НАЗНАЧЕНИЕ. Ввод данных с внешнего порта.
ФОРМАТ.

IN A

A — любое числовое выражение. Значение A, округленное до ближайшего целого, задает адрес порта и должно лежать в диапазоне от 0 до 65535.

ОПИСАНИЕ. Спектрум оперирует шестнадцатиразрядными адресами и может адресовать 65536 внешних портов. С помощью внешних портов в Спектруме адресуются клавиатура, магнитофон, джойстик и другие периферийные устройства. Причем, при выборе устройства не всегда происходит дешифрация всех шестнадцати разрядов адреса и, следовательно, устройство занимает $k \cdot 2^n$ адресов в адресном пространстве (k — число портов, используемых устройством; n — число разрядов адреса, не используемых при дешифрации).

В качестве примера рассмотрим способ считывания информации с клавиатуры. Клавиатура Спектрума состоит из сорока клавишей, разделенных на восемь групп по пять клавишей. Каждая из клавишей находится на пересечении восьми адресных шин A8÷A15 и пяти шин данных D0÷D4 в соответствии с рис.4.

	D0	D1	D2	D3	D4	
A15	space	SS	M	N	B	IN 32766
A14	enter	L	K	J	H	IN 49150
A13	P	O	I	U	Y	IN 57342
A12	0	9	8	7	6	IN 61438
A11	1	2	3	4	5	IN 63486
A10	Q	W	E	R	T	IN 64510
A09	A	S	D	F	G	IN 65022
A08	CS	Z	X	C	V	IN 65278

Рис.4. Схема клавиатуры Спектрума.

Если ни одна из клавиш не нажата, то по адресам клавиатуры считываются единицы. Нажатие на произвольную клавишу приводит к замыканию шины адреса на соответствующую шину данных.

Микрокомпьютер просматривает клавиатуру 50 раз в секунду. Делает он это следующим образом: генерирует поочередно адреса строк клавиатуры, начиная с первой строки (32766) до последней (65278) и проверяет информацию на шине данных. Нуль в одном из битов от D0 до D4 означает, что нажата определенная клавиша.

Проиллюстрируем вышеизложенное следующим примером. Введите и запустите следующую программу:

```
10 PRINT IN 63486: PAUSE 100
```

```
20 GO TO 10
```

Нажимая клавиши 1+5, убедитесь в соответствии получаемых кодов рис.4.

ПРИМЕЧАНИЕ. Биты D7, D5 безразличны для микрокомпьютера и могут быть 0, либо 1 в зависимости от схемотехники микрокомпьютера. Бит D6 используется для ввода информации с магнитофона.

3.35. Оператор INK.

НАЗНАЧЕНИЕ. Устанавливает цвет чернил.

ФОРМАТ.

INK N

где N — любое числовое выражение. Значение N, округленное до ближайшего целого, должно лежать в диапазоне от 0 до 9.

ОПИСАНИЕ. Оператор INK определяет цвет "чернил" изображения в знакоместе при выводе очередных символов на экран. Значения 0+7 задают код цвета, заносимого в байт поля атрибутов. Значение 8 определяет "прозрачный" цвет в том смысле, что информация о цвете "чернил" в соответствующем байте атрибутов не изменяется. Значение 9 задаст "контрастный" цвет. При задании цвета 9 автоматически устанавливается цвет "чернил", противоположный тому, в который установлена "бумага" (PAPER). Цвет "чернил" может быть задан глобально (для всех последующих вводимых символов) или локально, если ключевое слово INK входит в состав операторов PRINT, INPUT, PLOT и т.д. В последнем случае заданный цвет "чернил" актуален только в пределах действия одного оператора, в котором он был задан.

ПРИМЕР.

```
PAPER9: FOR i=0 TO 9: INK i: PRINT i: NEXT i
```

3.36. Функция INKEY\$.

НАЗНАЧЕНИЕ. Ввод символов с клавиатуры.

ФОРМАТ.

INKEY\$

ОПИСАНИЕ. Функция INKEY\$ выдает символ, введенный с клавиатуры. Если ни одна клавиша не нажата, выдается пустая строка. Функция INKEY\$ не ждет нажатия. Вводимые символы не отображаются на экране, а передаются в программу.

ПРИМЕР.

```
10 LET a$=INKEY$
20 IF a$=" " THEN GO TO 10
30 PRINT a$; : GO TO 10
```

3.37. Оператор INPUT.

НАЗНАЧЕНИЕ. Ввод данных с клавиатуры во время выполнения программы.

ФОРМАТ.

INPUT последовательность элементов

Элементом оператора INPUT может быть любой из следующих:

- номер канала;
- элемент оператора PRINT, не начинающийся с буквы (пусто, числовое выражение, строковое выражение, функция управления TAB);

— имя переменной;

— LINE (при этом обязательно имя строковой переменной). Элементы оператора INPUT могут быть разделены запятыми, точками с запятой и апострофами аналогично оператору PRINT.

ОПИСАНИЕ. Если элемент оператора INPUT начинается с буквы, то он должен быть переменной, значение которой необходимо ввести. Если элемент заключен в двойные кавычки, то он выводится в нижней части экрана в строке подсказки как строковая константа. В строке подсказки могут быть выведены также значения переменных. Для этого переменная должна быть заключена в скобки. При выводе в строке подсказки в операторе INPUT могут быть использованы функция управления AT и функция управления TAB.

Для ввода значения строковой переменной можно использовать ключевое слово LINE. При этом строковые кавычки не будут выводиться, как они обычно выводятся при вводе строковой переменной.

Прерывание оператора INPUT производится с помощью ключевого слова STOP в крайней левой позиции ввода. При этом выводится сообщение:

Н STOP in INPUT, n : k

Либо, если используется ключевое слово LINE, с помощью символа CS и 6.

ПРИМЕР.

```
10 LET m$=""
20 LET a=INT(RND*9)+1
30 LET b=INT(RND*9)+1
40 INPUT (m$) "(a); "; (b); "="; c
50 IF a*b=c THEN LET m$="right": GO TO 20
60 LET m$="wrong. try again": GO TO 40
```

Программа, приведенная в примере, проверяет знание таблицы умножения.

3.38. Функция INT.

НАЗНАЧЕНИЕ. Вычисление наибольшего целого, которое не превышает заданного значения.

ФОРМАТ.

INT X

где X — число, переменная, либо выражение в скобках.

ОПИСАНИЕ. Функция INT возвращает наибольшее число, которое не превышает значения аргумента.

ПРИМЕР.

```
PRINT INT 6.9
```

```
6
```

```
PRINT INT -4.5
```

```
-5
```

3.39. Оператор INVERSE.

НАЗНАЧЕНИЕ. Инверсия цвета.

ФОРМАТ.

```
INVERSE X
```

где X — любое числовое выражение.

ОПИСАНИЕ. Значение X, округленное до ближайшего целого, должно быть 0, либо 1, иначе выдается сообщение об ошибке. При выполнении оператора INVERSE с аргументом X=1 каждая точка знакоместа, выводимого на экран, изменит свой цвет таким образом, что цвет "бумаги" будет заменен на цвет "чернил", и наоборот. Оператор INVERSE с аргументом X=0 отменяет инверсию.

ПРИМЕР.

```
10 PRINT PAPER6; INK1; "PAPER yellow, INK blue"
```

```
RUN
```

```
INVERSE 1
```

```
GO TO 10
```

```
INVERSE 0
```

```
GO TO 10
```

3.40. Функция LEN.

НАЗНАЧЕНИЕ. Определение длины строки.

ФОРМАТ.

```
LEN A$
```

где A\$ — строковая константа, строковая переменная, либо строковое выражение в скобках.

ОПИСАНИЕ. Функция LEN возвращает число, равное количеству символов в строке.

ПРИМЕР.

```
10 LEN A$ = "string"
```

```
20 PRINT LEN A$
```


RUN

6

3.41. Оператор LET.

НАЗНАЧЕНИЕ. Оператор присваивания.

ФОРМАТ.

LET X=Y

X — имя переменной (элемента массива), которой должно быть присвоено значение;

Y — выражение любого типа.

ОПИСАНИЕ. Значение заданного выражения присваивается заданной переменной. Тип выражения (числовое или строковое) должен совпадать с типом переменной, в противном случае выдается сообщение об ошибке.

ПРИМЕР.

```
10 LET X=10
```

```
20 LET Y=X+5
```

```
30 LET A$="page"
```

```
40 PRINT X'Y'A$
```

```
RUN
```

```
10
```

```
15
```

```
page
```

3.42. Команда LIST.

НАЗНАЧЕНИЕ. Вывод программы на экран.

ФОРМАТ.

LIST [#S,][N]

S — номер канала;

N — номер строки программы;

N, S могут быть любыми числовыми выражениями.

ОПИСАНИЕ. Выполняется печать листинга (текста) программы на экране. В форме LIST N листинг выводится начиная со строки с номером равным, либо большим, чем значение N. Если текст занимает больше, чем один экран, то после заполнения экрана выдается запрос scroll? Если нажать клавиши BREAK, N, STOP или SPACE, то распечатка прервется. Нажатие любой другой клавиши продолжает вывод листинга.

ПРИМЕР.

```
10 LIST
20 LIST #1, 20
30 PAUSE 0
RUN
```

В приведенной в примере программе оператор 10 выводит весь листинг программы в верхней части экрана. Оператор 20 выводит текст программы начиная с оператора 20 в системном окне. Действие оператора 30 закончится при нажатии на любую клавишу, при этом в системном окне появится сообщение:

```
0 OK, 30:1
```

3.43. Команда LLIST.

НАЗНАЧЕНИЕ. Распечатка программы на принтере.

ФОРМАТ.

LLIST [#S, ##N]

S — номер канала;

N — номер строки программы;

N, S могут быть любыми числовыми выражениями.

ОПИСАНИЕ. Команда ZX-принтера. Выполняется так же, как и LIST, но распечатка выполняется на ZX-принтере.

3.44. Функция LN.

НАЗНАЧЕНИЕ. Вычисление натурального логарифма.

ФОРМАТ.

LN x

где x — число, переменная, либо выражение в скобках.

ОПИСАНИЕ. Функция вычисляет логарифм по основанию $e=2.7182818$. Для вычисления логарифма по любому другому основанию необходимо разделить натуральный логарифм числа на натуральный логарифм основания, т.е. $\log_a = \ln x / \ln a$.

ПРИМЕР.

```
10 DEF FN L(x)=LNx/LN10
20 PRINT FN L(100)
RUN
2
```


3.45. Команда LOAD.

НАЗНАЧЕНИЕ. Загрузка с магнитофона.

ФОРМАТ.

LOAD "[имя]" — загрузка Бейсик-программы.

LOAD "[имя]" CODE [A,N] — загрузка блока в машинных кодах.

A — начальный адрес загрузки;

N — число загружаемых байтов.

A, N могут быть любыми числовыми выражениями.

LOAD "[имя]" SCREEN\$

LOAD "[имя]" DATA A()

LOAD "[имя]" DATA A\$()

A() — имя числового массива;

A\$() — имя строкового массива.

ОПИСАНИЕ. Для ввода с магнитофона Бейсик-программ используется команда LOAD "имя". По этой команде вводится программа с именем, стоящим между двойными кавычками. Двойные кавычки в формате команды обязательны. Команда LOAD загружает новую программу с ленты, стирая все старые. Имя программы в команде может быть пустой строкой, в этом случае вводится первая попавшаяся Бейсик-программа. При вводе программы на экран выводится сообщение:

Program: имя

Для ввода с ленты последовательности байтов используется команда: LOAD "имя" CODE A,N.

Команда вводит N байтов с адреса A. Если A и (или) N опущены, то используются адрес и число байтов, указанные при выводе. Если байтов в массиве больше, чем указано в команде в параметре N, то выдается сообщение об ошибке:

R Tape loading error, n : k.

Двойные кавычки обязательны. Если уже существует массив байтов, обозначенный "имя", то при вводе командой LOAD кодового массива с тем же именем он уничтожается. Если "имя" — пустая строка, то вводится первый попавшийся массив байтов. При вводе кодового массива командой LOAD "имя" CODE на экран выводится сообщение:

Bytes : имя

Команда LOAD "имя" SCREEN\$ служит для ввода экранной области памяти, эквивалентна команде LOAD "имя" CODE 16384, 6912. На экран выводится сообщение:

Bytes : имя

Ввод с магнитной ленты числового, либо строкового массива осуществляется командами:

LOAD "имя" DATA A()

LOAD "имя" DATA A\$()

"Имя" может быть пустой строкой. В этом случае вводится первый попавшийся массив вида DATA. Если уже существует массив, обозначенный "имя", то при вводе командой LOAD массива с тем же именем старый массив уничтожается. При вводе массива DATA на экран выводится сообщение:

Number array : имя — в случае числового массива;

Character array : имя — в случае строкового массива.

3.46. Команда LPRINT.

НАЗНАЧЕНИЕ. Распечатка информации на принтере.

ФОРМАТ.

Аналогично команде PRINT.

ОПИСАНИЕ. Команда ZX-принтера. Выполняется также, как и PRINT, но распечатка выполняется на ZX-принтере.

3.47. Команда MERGE.

НАЗНАЧЕНИЕ. Ввод программы.

ФОРМАТ.

MERGE "имя"

ОПИСАНИЕ. Добавляет программу "имя" к некоторой программе, уже находящейся в памяти. Добавляются строки с номерами, которых нет в старой программе. Строки с совпадающими номерами заменяются новыми. Ввод массивов типа CODE и DATA не поддерживаются.

"Имя" может быть пустой строкой, при этом вводится первая попавшаяся программа.

Если для новой программы нет места, выдается сообщение:

4 Out of memory

3.48. Оператор MOVE.

НАЗНАЧЕНИЕ. Команда микродрайва.

ФОРМАТ.

MOVE "имя", "К"

имя — строковое выражение;

К — имя канала (одна буква).

ОПИСАНИЕ. Пересылка файла, обозначенного "имя", в канал с именем "К".

3.49. Команда NEW.

НАЗНАЧЕНИЕ. Установка в исходное состояние.

ФОРМАТ.

NEW

ОПИСАНИЕ. Уничтожает Бейсик-программу, устанавливает в исходное состояние системные переменные, очищает память до границы RAMTOP, установленной CLEAR. Эквивалентна по своему действию нажатию на кнопку RESET.

3.50. Оператор OPEN#.

НАЗНАЧЕНИЕ. Открытие канала.

ФОРМАТ.

OPEN#S, "К"

S — номер канала (любое числовое выражение).

К — имя канала (одна буква).

ОПИСАНИЕ. Связывает поток данных с номером, равным значению выражения S, которое должно лежать в диапазоне от 0 до 15, с каналом, обозначенным именем "К".

После включения питания (или нажатия на кнопку RESET) в фазе инициации операционной системы открываются потоки с номерами от 0 до 3 и связываются тем самым со следующими каналами:

канал	имя	
0	"К"	клавиатура
1	"К"	клавиатура
2	"S"	экран
3	"P"	принтер

Клавиатура "К" является единственным из вышеперечисленных каналов, который может транслировать данные в двух направлениях из компьютера и в него. Входным устройством служит клавиатура, а выходным — нижняя часть экрана (системное окно).

Каналы "P" и "S" могут быть использованы только в качестве выходных устройств.

В Спектруме стандартным способом можно использовать только три канала. В ПЗУ компьютера находятся готовые процедуры, позволяющие работать с клавиатурой, экраном и принтером.

В ПЗУ системы Интерфейс I имеются процедуры обслуживания микродрайва. Если вы хотите подключить к Спектруму нестандартное устройство, то необходимо запрограммировать соответствующий канал.

ПРИМЕР.

```
10 OPEN#6, "S"
20 PRINT#6; "screen"
30 PAUSE 50
40 CLOSE#6
50 OPEN#6, "K"
60 PRINT#6; "keybord"
70 PAUSE 100
```

3.51 Оператор OUT.

НАЗНАЧЕНИЕ. Пересылка байта данных в заданный порт микрокомпьютера.

ФОРМАТ.

OUT A, X

A — любое числовое выражение. Значение A, округленное до ближайшего целого, задает адрес порта и должно лежать в диапазоне от 0 до 65535;

X — любое числовое выражение. Абсолютная величина значения X, округленная до ближайшего целого, должна лежать в диапазоне от 0 до 255.

ОПИСАНИЕ. Операнд X определяет числовое значение передаваемого байта данных. Этот байт пересылается в порт микрокомпьютера, адрес которого указан в операнде A.

Значения битов порта вывода связано со значением X уравнением: $128D7+64D6+32D5+16D4+8D3+4D2+2D1+1D0=INT(ABS(X+0.5))$

Из выходных портов, используемых Спектрумом, отметим порт с адресом 254. Назначение битов этого порта:

D4 — обеспечивает работу с громкоговорителем;

D3 — вывод на магнитофон;

D2, D1, D0 — определяет цвет бордюра.

ПРИМЕР.

```
10 FOR i=0 TO 7
20 OUT 254, i: PAUSE 50
30 NEXT i
RUN
```

3.52. Оператор OVER.

НАЗНАЧЕНИЕ. Задаст наложение символов.

ФОРМАТ.

OVER N

N — любое числовое выражение, значение которого, округленное до ближайшего целого, должно принимать значение 0, либо 1.

ОПИСАНИЕ. Оператор OVER1 включает режим наложения, а оператор OVER0 выключает его. При включенном режиме наложения, если какой-либо знак записывается на уже занятое место, то не происходит удаление старого знака, а только нанесение тех точек, которые имеют в новом знаке цвет "чернил". Места, в которых точки в цвете "чернил", наложенных друг на друга знаков, перекрываются, принимают цвет фона.

ПРИМЕР.

```
10 PRINT AT10, 15; "X";
20 PRINT CHR$8;: PAUSE 100
30 OVER1
40 PRINT "0"
RUN
```

В результате выполнения этой программы на экран будет выведен знак, являющийся суммой (в определенном выше смысле) знаков X и 0. Выполнение оператора PRINT CHR\$8 равносильно возврату на одну позицию влево.

3.53. Оператор PAPER.

НАЗНАЧЕНИЕ. Задание цвета фона.

ФОРМАТ.

PAPER N

N — любое числовое выражение, значение которого, округленное до ближайшего целого, должно лежать в диапазоне от 0 до 9.

ОПИСАНИЕ. Оператор PAPER определяет цвет фона изображений в знакоместе при выводе очередного символа на экран. Значения

0+7 задают код цвета, заносимого в байт поля атрибутов. Значение 8 определяет "прозрачный" цвет в том смысле, что информация о цвете фона в соответствующем байте атрибутов не изменяется. Значение 9 задает "контрастный" цвет. При задании цвета 9 автоматически устанавливается цвет фона противоположный тому, в который установлены "чернила" (INK).

Цвет фона может быть задан глобально (для всех последующих вводимых символов) или локально, если ключевое слово PAPER входит в состав операторов PRINT, INPUT, PLOT и т.д. В последнем случае заданный цвет фона актуален только в пределах действия одного оператора, в котором он был задан.

П Р И М Е Р .

```
INK9: FOR P=0 TO 9: PAPER P: PRINT P:
NEXT P
```

3.54. Оператор PAUSE.

НАЗНАЧЕНИЕ. Установка временной задержки.

ФОРМАТ.

PAUSE N

N — любое числовое выражение.

ОПИСАНИЕ. Значение N, округленное до ближайшего целого, должно лежать в диапазоне от 0 до 65535. Приостанавливает выполнение программы и отображает картину на экране в течение N кадров (N/50 сек.). Максимальное значение N дает задержку до 22 мин. Значение N=0 дает паузу бесконечной продолжительности. Паузу всегда можно прервать, нажав на любую клавишу.

П Р И М Е Р .

```
10 REM program clock
20 INPUT "hours?"; hh, "minutes?"; mm: CLS
30 FOR h=hh TO 23
40 FOR m=mm TO 59
50 FOR S=0 TO 59
60 LET h1=INT(h/10): LET m1=INT(m/10): LET S1=
INT(S/10)
70 PRINT AT 10, 12; h1; h-10*h1; ":"; AT 10, 15; m1;
m-10*m1; ":"; AT 10, 18; S1; S-10*S1
80 PAUSE 46: NEXT S
90 NEXT m: LET mm=0
100 NEXT h: LET hh=0: GO TO 30
```


В этом примере приведена программа, моделирующая часы. Ввод текущего времени осуществляется оператором 20. "Маятником" этих часов является оператор PAUSE 46. Казалось бы, здесь должен быть оператор PAUSE 50 для точного счета одной секунды, но тогда не было бы учтено время, затрачиваемое на выполнение остальных операторов программы. Приведенная программа обеспечивает точность "хода" часов порядка 0,2% или, иными словами, уход на три минуты в сутки.

3.55. Функция PEEK.

НАЗНАЧЕНИЕ. Выборка байта данных из памяти.
ФОРМАТ.

PEEK A

A — число, переменная, либо выражение в скобках.

ОПИСАНИЕ. Результатом выполнения функции PEEK является целое число в диапазоне от 0 до 255, которое представляет собой значение байта памяти, находящегося по адресу, указанному A. Значение A, округленное до ближайшего целого, должно лежать в диапазоне от 0 до 65535, и определяет адрес, по которому выбирается значение байта.

ПРИМЕР.

```
10 LET FRAMES=65536*PEEK23674+256*PEEK23673+
PEEK23672
20 PRINT FRAMES/50
RUN
```

В примере выводится на экран деленное на 50 значение системной переменной FRAMES. Переменная FRAMES является счетчиком кадров и наращивается на единицу каждые 20мс, таким образом, на экран выводится количество секунд, прошедшее с момента включения микрокомпьютера.

3.56. Функция PI.

НАЗНАЧЕНИЕ. Задание числа $\pi = 3.1415927$.
ФОРМАТ.

PI

ПРИМЕР.

```
PRINT PI
3.1415927
```

3.57. Оператор PLOT.

НАЗНАЧЕНИЕ. Вывод точки на экран.

ФОРМАТ.

PLOT [упр1, упр2...;] X, Y

упр1, упр2... — управляющие символы INK, PAPER, FLASH, BRIGHT, OVER, INVERSE;

X — любое числовое выражение;

Y — любое числовое выражение.

ОПИСАНИЕ. Оператор осуществляет вывод на экран пикселя (точки) с координатами X, Y. Значения выражений X, Y, округленные до ближайшего целого, должны лежать в диапазоне:

X — от 0 до 255;

Y — от 0 до 175.

Локальные атрибуты знакомеда, в которое выводится точка (цвет "чернил", фона, яркость, мерцание), задаются управляющими символами INK, PAPER, BRIGHT, FLASH.

PLOT INVERSE1 — заданный пиксель принимает цвет фона.

PLOT OVER1 — выводит заданный пиксель в режиме "наложения". Операторы PLOT INVERSE и PLOT OVER устанавливают цвет "чернил" в знакоместе, равный глобальному, в соответствии с системной переменной ATTR P(23693).

П Р И М Е Р .

```
10 FOR P=0 TO 9
```

```
20 FOR i=0 TO 9
```

```
30 PLOT BRIGHT1; PAPER P; INK i; P*8+3, i*8+3
```

```
40 NEXT i
```

```
50 NEXT P
```

```
RUN
```

В примере демонстрируется вывод точек с помощью оператора PLOT при всех возможных комбинациях цвета, фона и чернил.

3.58. Функция POINT.

НАЗНАЧЕНИЕ. Проверка состояния точки экрана.

ФОРМАТ.

POINT (X, Y)

X — любое числовое выражение;

Y — любое числовое выражение.

ОПИСАНИЕ. Функция POINT принимает значение 0, если точка с координатами X, Y имеет цвет фона, и значение 1, если цвет "чернил". Значение выражения X, округленное до ближайшего целого, должно лежать в диапазоне от 0 до 255, значение Y в диапазоне от 0 до 175.

ПРИМЕР.

```
CLS: PRINT POINT(100, 100)
```

```
0
```

3.59. Оператор POKE.

НАЗНАЧЕНИЕ. Пересылка байта данных в память.
ФОРМАТ.

```
POKE A, N
```

A — любое числовое выражение, значение которого является адресом засылки;

N — любое числовое выражение, значение которого засылается по указанному адресу.

ОПИСАНИЕ. Значение A, округленное до ближайшего целого, должно лежать в диапазоне от 0 до 65535, иначе выдается сообщение об ошибке. Значение N, округленное до ближайшего целого, должно лежать в диапазоне от -255 до 255.

ПРИМЕР.

```
10 FOR N= -255 TO 255
```

```
20 POKE 40000, N
```

```
30 PRINT N, PEEK 40000
```

```
40 NEXT N
```

3.60. Оператор PRINT.

НАЗНАЧЕНИЕ. Вывод информации на экран.

ФОРМАТ.

```
PRINT [последовательность элементов ][:];[I, I']
```

Элементом оператора PRINT может быть любой из следующих:

#S — номер канала, где S — любое числовое выражение;

упр — управляющие символы INK, PAPER, FLASH, BRIGHT, OVER, INVERSE;

АТ М, N — функция управления, задающая печать с заданной позиции, где М, N — любые числовые выражения, значения которых определяют соответственно номер строки и столбца;

ТАВ К — функция табуляции, задающая печать с табуляцией, где К — любое числовое выражение, значение которого задает позицию табуляции;

выражение — любое числовое или строковое выражение.

Элементы в операторе PRINT разделяются запятыми (,), точками с запятой (;) или апострофами (').

ОПИСАНИЕ. Оператор PRINT осуществляет вывод списка значений числовых и (или) строковых выражений на экран дисплея, либо на устройство, подключенное к каналу с указанным номером. Если список выражений не указан, то происходит пропуск одной строки. Строковые константы должны быть заключены в кавычки. Локальные цветовые атрибуты выводимых символов могут быть заданы с помощью управляющих символов INK, PAPER, BRIGHT, FLASH, INVERSE. Их действие аналогично действию соответствующих операторов, но распространяется только в пределах того оператора PRINT, в котором они использованы.

Позиция каждого выводимого элемента определяется функциями АТ или ТАВ и (или) знаками, используемыми для разделения элементов списка.

Точка с запятой (;) вызывает вывод следующего элемента непосредственно со следующей позиции.

Запятая (,) в списке выражений заставляет печататься следующее значение с начала следующей зоны (строка Спектрума разбита на две зоны по шестнадцать позиций).

Апостроф (') используется для вывода с начала следующей строки.

Если запятая или точка с запятой заканчивают список выражений, то следующий оператор PRINT начинает печать в той же строке с позиции, определяемой разделителем. Если длина выводимой строки больше 32 позиций, то печать продолжается на следующей физической строке.

Значения выражений М и N в функции АТ должны лежать в диапазоне: М — от 0 до 21; N — от 0 до 31.

Значения выражения К в функции ТАВ должно лежать в диапазоне от 0 до 65535. Значение К указывает на номер столбца, с которого должен осуществляться вывод. Если значение К больше 32, то используется остаток от деления К на 32. Так PRINT ТАВ 33 равнозначно PRINT АТ1.

Если выводимая информация не помещается на один экран, то происходит его свертка. Когда один экран заполнится, то вывод приостановится, и в нижней части экрана появится запрос:
scrool?

Для вывода следующей страницы нужно нажать любую клавишу, кроме BREAK, N, SPACE, либо STOP (CS и A). В последнем случае компьютер остановит выполнение программы и выведет сообщение:

D BREAK — CONT repeats, n : k

Продолжить вывод можно с помощью команды CONTINUE.

ПРИМЕР.

```
10 PRINT AT 0, 16; 1
20 PRINT AT 1, 12; "CONTENTS"; AT 2, 24; "PAGE"
30 FOR i=1 TO 40
40 PRINT TAB3; "CHAPTER"; TAB 11; i, "....."
50 NEXT i
```

Эта программа выводит на экран оглавление для первой страницы книги.

3.61 Оператор RANDOMIZE.

НАЗНАЧЕНИЕ. Переустановка базы генератора случайных чисел.
ФОРМАТ.

RANDOMIZE [N]

N — любое числовое выражение, значение которого должно лежать в диапазоне от 0 до 65535.

ОПИСАНИЕ. Интерпретатор Бейсика содержит генератор случайных чисел, последовательность которых можно получить, используя функцию RND (п.3.66). На самом деле получаемые числа являются псевдослучайными. Их последовательность генерируется с использованием системной переменной SEED, являющейся базой генератора случайных чисел, и будет одной и той же при различных выполнениях программы, если база генерации остается неизменной.

Оператор RANDOMIZE позволяет переустановить системную переменную SEED и тем самым изменить последовательность генерируемых чисел. При выборе базы генерации для увеличения элемента случайности можно, например, использовать значение системной переменной FRAMES.

ПРИМЕР.

```
10 RANDOMIZE (256*PEEK 23673 + PEEK 23672)
20 PRINT INT(10*RND)
RUN
```

Приведенная в примере программа печатает случайное целое число в диапазоне от 0 до 9.

3.62. Оператор READ.

НАЗНАЧЕНИЕ. Присвоение переменным значений, заданных в операторах DATA.

ФОРМАТ.

READ X [Y]...

X, Y — числовые или строковые переменные, элементы массивов.

ОПИСАНИЕ. Оператор READ всегда используется в сочетании с оператором DATA (п.3.21). Значения, заданные в операторе DATA, присваиваются переменным, перечисленным в операторе READ в порядке их перечисления один к одному. Первой переменной присваивается первое значение, второй переменной — второе и т.д. Один оператор READ может получать значения, заданные в нескольких операторах DATA (операторы DATA обрабатываются в порядке их появления в программе). В то же время несколько операторов READ могут обращаться к одному оператору DATA. Если количество переменных в операторе READ больше количества значений в операторе DATA, то выдается сообщение об ошибке:

E Out of DATA, n : k

Если количество переменных меньше, чем количество значений, заданных в операторах DATA, то последующий оператор READ будет выбирать значения, начиная с первого еще не прочитанного значения данных.

В Бейсике существует возможность указать оператор DATA, начиная с которого последующие операторы READ будут выбирать значения для присваивания их переменным. Для этой цели служит оператор RESTORE (п.3.64), который позволяет также повторно выбирать уже использованные данные.

Тип переменной должен соответствовать типу присваиваемого значения. Иначе выдается сообщение об ошибке.

ПРИМЕР.

```
10 PRINT "country"; TAB 10; "city"; TAB 24 "index"
20 READ C$, S$, N
30 DATA "USSR", "Rostov-on-Don", 344000
40 PRINT C$; TAB 10; S$; TAB 24; N
RUN
country      city              index
USSR —      Rostov-on-Don    344000
```

3.63 Оператор REM.

НАЗНАЧЕНИЕ. Служит для внесения в текст программы комментариев.

ФОРМАТ.

REM [комментарий]

комментарий — любая последовательность символов Спектрума.

ОПИСАНИЕ. Оператор REM является невыполнимым оператором. Присутствуя в тексте программы, оператор REM замедляет скорость интерпретации программы и увеличивает занимаемую ею память. Оператору REM может быть передано управление с помощью операторов GO TO и GO SUB. В этом случае выполнение программы продолжается с первого выполняемого оператора, следующего за оператором REM.

Если комментарий помещается в одну строку с другими операторами Бейсика, то оператор REM должен быть последним оператором в строке.

3.64. Оператор RESTORE.

НАЗНАЧЕНИЕ. Переустановка указателя оператора DATA.

ФОРМАТ.

RESTORE [N]

N — любое числовое выражение, значение которого, округленное до ближайшего целого, задает номер строки оператора DATA.

ОПИСАНИЕ. Если в операторе RESTORE не задан номер строки, то следующий оператор READ будет обращаться к первому элементу первого оператора DATA. Если в операторе RESTORE задан номер строки, то следующий оператор READ будет обращаться к первому элементу оператора READ, стоящему в данной строке или после нее.

ВНИМАНИЕ! Диапазон значений N, проверяемый компьютером на допустимость, в операторе RESTORE равен 0÷65535, а не 1÷9999. При некоторых значениях N, например 65535, возможно зависание компьютера.

ПРИМЕР.

```
10 READ a, b, c
20 RESTORE
30 READ x, y
40 PRINT a; b; c; x; y
50 DATA 1, 2, 3
RUN
12312
```

3.65. Оператор RETURN.

НАЗНАЧЕНИЕ. Возврат из подпрограммы.

ФОРМАТ.

RETURN

ОПИСАНИЕ. Смотри п.3.31. Операторы GO SUB и RETURN.

3.66. Функция RND.

НАЗНАЧЕНИЕ. Генерация псевдослучайных чисел.

ФОРМАТ.

RND

ОПИСАНИЕ. Функция возвращает псевдослучайное число в диапазоне $0 \leq \text{RND} < 1$

Для получения целых случайных чисел в диапазоне от 0 до n можно воспользоваться формулой:

$\text{INT}(\text{RND} * (n+1))$

ПРИМЕР.

```
10 BORDER 5: PAPER 6: CLS
20 LET X=120: LET Y=88: PLOT X, Y
30 LET dX=10-INT(RND*21)
40 IF X+dX>=255 OR X+dX<=0 THEN GO TO 30
50 LET X=X+dX
60 LET dY=7-INT(RND*15)
70 IF Y+dY>=175 OR Y+dY<=0 THEN GO TO 60
80 LET Y=Y+dY
90 DRAW dX, dY: GO TO 30
```


Приведенная программа может служить для иллюстрации Броуновского движения.

3.67. Команда RUN.

НАЗНАЧЕНИЕ. Команда на исполнение программы.
ФОРМАТ.

RUN [N]

N — номер строки (допустимо любое числовое выражение, значение которого лежит в диапазоне от 0 до 9999).

ОПИСАНИЕ. Команда RUN начинает выполнение программы, находящейся в памяти. Если задан номер строки, то выполнение начинается с указанной строки. Если номер не указан, то со строки, имеющей наименьший номер.

Команда RUN уничтожает все переменные, имевшиеся до нее, и очищает экран.

3.68. Команда SAVE.

НАЗНАЧЕНИЕ. Запись программы или данных на магнитофон.
ФОРМАТ.

SAVE "имя" [LINE N] — запись Бейсик-программы с автозапуском после загрузки со строки с номером N.

SAVE "имя" CODE M, N — запись на ленту блока машинных кодов, начиная с адреса M, длиной в N байтов.

SAVE "имя" SCREEN\$ — запись экрана.

SAVE "имя" DATA A() — запись на ленту числового массива с именем A.

SAVE "имя" DATA A\$() — запись на ленту строкового массива с именем A.

"имя" — наименование файла при записи на магнитную ленту.

ОПИСАНИЕ. Имя файла не может быть пустым. Оно может содержать до 10 любых символов Спектрума и может начинаться с любого символа. Строчные и прописные буквы воспринимаются как различные символы.

В команде SAVE LINE допустимо задание номера строки автозапуска с помощью числового выражения.

В команде SAVE CODE параметры M, N могут быть любыми числовыми выражениями.

Команда `SAVE SCREEN$` является особой формой вывода на магнитофон экранной области памяти.

Скобки в командах `SAVE DATA ( )` и `SAVE DATA $( )` обязательны. Команда `SAVE` всегда выдает подсказку:

Start tape, then press any key.

(запусти магнитофон и нажми любую клавишу).

Команда ждет нажатия, после чего начинает запись информации магнитофон.

3.69. Функция `SCREEN$`.

НАЗНАЧЕНИЕ. Предназначена для чтения символа в заданном месте экрана.

ФОРМАТ.

`SCREEN$(N, M)`

N — номер строки; M — номер столбца.

N, M — любые числовые выражения.

ОПИСАНИЕ. Возвращает символ, равный тому, который находится на экране в знакоместе с координатами N, M. Функция не может считывать знаки, определенные пользователем, графические символы, а также строки, нарисованные с помощью `PLOT`, `DRAW` и `CIRCLE`. Попытка считать вышеуказанные символы приведет к выводу пустой строки.

Значения выражений N, M должны лежать в диапазоне:

N — от 0 до 23, M — от 0 до 31.

ПРИМЕР.

`PRINT AT 0, 0; SCREEN$(11, 16)`

По этой команде будет считан символ, находящийся на пересечении 11-й строки и 16-го столбца экрана, и он же будет выведен в верхнем левом углу экрана.

3.70. Функция `SGN`.

НАЗНАЧЕНИЕ. Определение знака числа.

ФОРМАТ.

`SGN X`

X — число, переменная, либо выражение в скобках.

ОПИСАНИЕ. Функция SGN (сигнум) задается следующим образом:

SGN X = 1, если $X > 0$

SGN X = 0, если $X = 0$

SGN X = -1, если $X < 0$

ПРИМЕР.

```
PRINT SGN 22
```

```
1
```

```
PRINT SGN (3-3)
```

```
0
```

```
PRINT SGN -0.5
```

```
-1
```

3.71. Функция SIN.

НАЗНАЧЕНИЕ. Вычисление синуса заданного угла.

ФОРМАТ.

SIN X

X — число, переменная, либо выражение в скобках.

ОПИСАНИЕ. Функция вычисляет синус угла, значение которого в радианах равно X. Для преобразования значения угла из градусной меры в радианную нужно значение угла умножить на $\pi / 180$, где $\pi = 3,1415927$.

ПРИМЕР.

```
10 LET A=30
```

```
20 LET X=A*PI/180
```

```
30 PRINT SIN X
```

```
RUN
```

```
0.5
```

В этом примере вычисляется синус угла A, равного 30 градусам.

3.72. Функция SQR.

НАЗНАЧЕНИЕ. Вычисление квадратного корня из заданного числа.

ФОРМАТ.

SQR X

X — число, переменная, либо выражение в скобках.

ОПИСАНИЕ. Значением X может быть любое неотрицательное число.

ПРИМЕР.

```
PRINT SQR 25
```

```
5
```

3.73. Оператор STOP.

НАЗНАЧЕНИЕ. Завершение выполнения программы.
ФОРМАТ.

STOP

ОПИСАНИЕ. Оператор STOP может быть указан в любом месте программы. В результате выполнения этого оператора на экран выводится сообщение:

9 STOP statement, n : k

где n — номер строки, в которой встретился оператор STOP, а k — номер оператора в строке.

В точке останова можно выполнить различные отладочные действия, например, проверить или изменить значение переменных. Выполнение программы можно продолжить с помощью оператора CONTINUE.

ПРИМЕР.

```
10 INPUT "X="; X, "Y="; Y
20 LET S=X+Y: STOP
30 PRINT S/100
RUN
X=5          Y=8
9 STOP statement, 20 : 2
PRINT S
13 CONTINUE
0.13
```

В этом примере после ввода значений переменных X и Y вычисляется значение переменной S и происходит останов в строке 20. В точке останова проверяется значение переменной S и выполнение программы возобновляется с помощью оператора CONTINUE.

3.74. Функция STR\$.

НАЗНАЧЕНИЕ. Преобразование числа в строку символов.
ФОРМАТ.

STR\$ X

X — число, переменная, либо выражение в скобках.

ОПИСАНИЕ. Результатом выполнения функции является строка символов, представляющая значение аргумента функции. Перед отрицательными числами выводится знак минус.

ПРИМЕР.

```
PRINT STR$ -1e3
```

```
- 1000
```

3.75. Функция TAN.

НАЗНАЧЕНИЕ. Вычисление тангенса заданного угла.

ФОРМАТ.

TAN X

X — число, переменная, либо выражение в скобках.

ОПИСАНИЕ. Функция вычисляет тангенс угла, значение которого в радианах равно X. Для преобразования значения угла из градусной меры в радианную нужно значение угла умножить на $\pi / 180$, где $\pi = 3,1415927$.

ПРИМЕР.

```
10 INPUT "A?"; a
```

```
20 LET X=a*PI/180
```

```
30 PRINT "TAN"; a; "="; TAN X
```

```
40 GO TO 10
```

Приведенная программа вычисляет тангенс любого заданного числа.

3.76 ФункцияUSR.

НАЗНАЧЕНИЕ. Вызов подпрограммы в машинном коде.

ФОРМАТ.

USR A

A — число, переменная, либо выражение в скобках.

USR A\$

A\$ — строковая константа, строковая переменная, либо строковое выражение в скобках.

ОПИСАНИЕ. Функция с числовым аргументом служит для вызова из Бейсика подпрограммы в машинных кодах. Значение аргумента определяет начальный адрес, с которого запускается подпрограмма. Для возвращения в Бейсик подпрограмма в машинных кодах должна заканчиваться машинной инструкцией RET (код 201). USR является функцией, и поэтому ее нельзя использовать самостоятельно в строке программы. Она должна входить в состав операторов, например, LET переменная = USR A, PRINT USR A и т.д. Значением функции USR после завершения действия подпрограммы в машинных кодах является содержимое пары регистров процессора Z80 BC. Не всегда, однако, этот результат бывает нужен. В этом случае часто для вызова подпрограммы в ма-

шинных кодах используется оператор:

RANDOMIZE USR A

В этом случае содержимое пары регистров BC служит только для переустановки базы генератора случайных чисел.

ПРИМЕР.

10 example of USR function

20 RANDOMIZE USR 4757

RUN

© 1982 Sinclair Research Ltd

В этом примере с помощью оператора 20 происходит обращение к системной процедуре, выводящей начальное сообщение Спектрума. Однако, обнуление памяти не происходит, в чем можно убедиться, нажав на клавишу ENTER.

Функция USR со строковым аргументом дает адрес в области графики пользователя UDG, в котором расположено изображение символа графики пользователя, соответствующего значению аргумента функции. Значением аргумента функции USR A\$ может быть только один символ от A до U. Область графики пользователя UDG занимает 168 байтов памяти с адресами от 65388 до 65535. Так как каждый знак Спектрума определяется восемью байтами (матрицей 8*8 точек), в области UDG можно определить 21 собственный графический символ. После включения питания операционная система Спектрума вписывает в область определяемой графики большие буквы от A до U.

ПРИМЕР.

10 FOR i=0 TO 7

20 READ n: POKE USR "u"+i, n

30 NEXT i

40 DATA 0, 76, 82, 114, 82, 82, 76, 0

В этом примере символ графики пользователя U в графическом режиме определяется как русская буква Ю в соответствии с рис.5.

двоичный код

десятич-
ный код

0	0	0	0	0	0	0	0	0
0	1	0	0	1	1	0	0	
0	1	0	1	0	0	1	0	
0	1	1	1	0	0	1	0	
0	1	0	1	0		1	0	
0	1	0	1	0	0	1	0	
0	1	0	0	1	1	0	0	
0	0	0	0	0	0	0	0	

0

76

82

114

82

82

76

0

Рис.5 Пример определяемой графики (буквы Ю).

Аналогично можно определить остальные буквы русского алфавита, не совпадающие по начертанию с латинскими.

3.77. Функция VAL.

НАЗНАЧЕНИЕ. Преобразование числа из символьной формы представления в числовую.

ФОРМАТ.

VAL A\$

A\$ — строка, строковая переменная, либо строковое выражение в скобках.

ОПИСАНИЕ. Функция определяет числовое значение строки A\$. Вычисление выполняется в два приема: сначала аргумент функции вычисляется как строка, затем кавычки отбрасываются и оставшееся выражение вычисляется как число. Незначащие пробелы при вычислении убираются.

ПРИМЕР.

```
PRINT VAL("-4"+"*5")  
-20
```

3.78. Функция VAL\$.

НАЗНАЧЕНИЕ. Обработка строки как строкового выражения.

ФОРМАТ.

VAL\$ "A\$"

A\$ — строковое выражение.

ОПИСАНИЕ. Функция VAL\$ так же, как и функция VAL вычисляется в два приема: сначала ее аргумент вычисляется как строка, затем кавычки отбрасываются и оставшаяся величина снова вычисляется как строка.

Следует помнить, что в Бейсике Спектрума внутри строки символ кавычек должен быть введен дважды. Для более глубокого вложения строк кавычки потребуется повторить четыре или восемь раз.

Если аргумент функции содержит синтаксическую ошибку или дает числовое значение, то появляется сообщение:

C Nonsense in BASIC n ; k

ПРИМЕР.

```
10 LET a$="stroka"  
20 PRINT VAL$ "a$"  
30 PRINT VAL$ ""a$""  
RUN  
stroka  
a$
```

3.79. Команда VERIFY.

НАЗНАЧЕНИЕ. Сравнение программы и данных, считываемых с магнитофона, с программами и данными в памяти компьютера.

ФОРМАТ.

```
VERIFY "[имя]"  
VERIFY "[имя]" CODE [M, N]  
VERIFY "[имя]" DATA A( )  
VERIFY "[имя]" DATA A$( )  
VERIFY "[имя]" SCREEN$
```

ОПИСАНИЕ. Команда VERIFY аналогична команде LOAD, но в отличие от нее не производит загрузку в память, а только сверяет то, что записано на ленте с тем, что находится в памяти компьютера, и в случае несоответствия выдает сообщение об ошибке. Имеет все те же форматы, что и оператор LOAD (п.3.45).

ПРИЛОЖЕНИЕ 1.
Полный набор символов «Спектрума».

десятичный код	шестнадцатиричный код	Символ
0	1	Не используется
1	1	Не используется
2	2	Не используется
3	3	Не используется
4	4	TRUE VIDEO
5	5	INV. VIDEO
6	6	CAPS LOCK
7	7	EDIT
8	8	← (курсор влево)
9	9	→ (курсор вправо)
10	A	↓ (курсор вниз)
11	B	↑ (курсор вверх)
12	C	DELETE
13	D	ENTER
14	E	Курсор E (переход в расширенный режим)
15	F	GRAPHICS
16	10	Чернила упр.
17	11	Фон. упр.
18	12	Мерцание упр.
19	13	Повышенная яркость упр.
20	14	Инверсия упр.

десятичный код	шестнадцатиричный код	Символ
21	15	Наложение упр.
22	16	Функция АТ упр.
23	17	Табуляция упр.
24	18	Не используется
25	19	Не используется
26	1A	Не используется
27	1B	Не используется
28	1C	Не используется
29	1D	Не используется
30	1E	Не используется
31	1F	Не используется
32	20	Пробел
33	21	! (восклицательный знак)
34	22	" (двойные кавычки)
35	23	# (знак номера или диз)
36	24	\$ (знак денежной единицы)
37	25	% (процент)
38	26	& (амперсанд)
39	27	' (апостроф)
40	28	((левая круглая скобка)
41	29	) (правая круглая скобка)
42	2A	* (умножение или звездочка)
43	2B	+ (плюс)
44	2C	, (запятая)
45	2D	- (минус)
46	2E	. (точка)
47	2F	/ (деление или косая черта)
48	30	0
49	31	1
50	32	2
51	33	3
52	34	4

десятичный код	шестнадцатиричный код	Символ
53	35	5
54	36	6
55	37	7
56	38	8
57	39	9
58	3A	: (двосточие)
59	3B	; (точка с запятой)
60	3C	< (меньше)
61	3D	= (равно)
62	3E	> (больше)
63	3F	? (вопросительный знак)
64	40	@ (коммерческая "а")
65	41	A
66	42	B
67	43	C
68	44	D
69	45	E
70	46	F
71	47	G
72	48	H
73	49	I
74	4A	J
75	4B	K
76	4C	L
77	4D	M
78	4E	N
79	4F	O
80	50	P
81	51	Q
82	52	R
83	53	S
84	54	T

Продолжение приложения 1.

десятичный код	шестнадцатиричный код	Символ
85	55	U
86	56	V
87	57	W
88	58	X
89	59	Y
90	5A	Z
91	5B	[(левая квадратная скобка)
92	5C	\ (обратная косая черта)
93	5D	] (правая квадратная скобка)
94	5E	↑ (возведение в степень)
95	5F	— (подчеркивание)
96	60	£ (фунт стерлингов)
97	61	a
98	62	b
99	63	c
100	64	d
101	65	e
102	66	f
103	67	g
104	68	h
105	69	i
106	6A	j
107	6B	k
108	6C	l
109	6D	m
110	6E	n
111	6F	o
112	70	p
113	71	q
114	72	r
115	73	s
116	74	t

десятичный код	шестнадцатиричный код	Символ
117	75	u
118	76	v
119	77	w
120	78	x
121	79	y
122	7A	z
123	7B	{ (левая фигурная скобка)
124	7C	(вертикальная черта)
125	7D	} (правая фигурная скобка)
126	7E	~ (знак Тильды)
127	7F	©
128	80	
129	81	
130	82	
131	83	
132	84	
133	85	
134	86	
135	87	
136	88	
137	89	
138	8A	
139	8B	
140	8C	
141	8D	
142	8E	
143	8F	
144	90	Λ (символ графики пользователя)
145	91	В (символ графики пользователя)
146	92	С (символ графики пользователя)
147	93	D (символ графики пользователя)
148	94	E (символ графики пользователя)

десятичный код	шестнадцатичный код	Символ
149	95	Г (символ графики пользователя)
150	96	G (символ графики пользователя)
151	97	H (символ графики пользователя)
152	98	I (символ графики пользователя)
153	99	J (символ графики пользователя)
154	9A	K (символ графики пользователя)
155	9B	L (символ графики пользователя)
156	9C	M (символ графики пользователя)
157	9D	N (символ графики пользователя)
158	9E	O (символ графики пользователя)
159	9F	P (символ графики пользователя)
160	A0	Q (символ графики пользователя)
161	A1	R (символ графики пользователя)
162	A2	S (символ графики пользователя)
163	A3	T (символ графики пользователя)
164	A4	U (символ графики пользователя)
165	A5	RND
166	A6	INKEY\$
167	A7	PI
168	A8	FN
169	A9	POINT
170	AA	SCREEN\$
171	AB	ATTR
172	AC	AT
173	AD	TAB
174	AE	VAL\$
175	AF	CODE
176	BO	VAL
177	B1	LEN
178	B2	SIN
179	B3	COS
180	B4	TAN
181	B5	ASN

Продолжение приложения 1.

десятичный код	шестнадцатиричный код	Символ
182	B6	ACS
183	B7	ATN
184	B8	LN
185	B9	EXP
186	BA	INT
187	BB	SQR
188	BC	SGN
189	BD	ABS
190	BE	PEEK
191	BF	IN
192	CO	USR
193	C1	STR\$
194	C2	CHR\$
195	C3	NOT
196	C4	BIN
197	C5	OR
198	C6	AND
199	C7	<= (меньше или равно)
200	C8	>= (больше или равно)
201	C9	<> (не равно)
202	CA	LINE
203	CB	THEN
204	CC	TO
205	CD	STEP
206	CE	DEF FN
207	CF	CAT
208	DO	FORMAT
209	D1	MOVE
210	D2	ERASE
211	D3	OPEN#
212	D4	CLOSE#
213	D5	MERGE

десятичный код	шестнадцатиричный код	Символ
214	D6	VERIFY
215	D7	BEEP
216	D8	CIRCLE
217	D9	INK
218	DA	PAPER
219	DB	FLASH
220	DC	BRIGHT
221	DD	INVERSE
222	DE	OVER
223	DF	OUT
224	EO	LPRINT
225	E1	LLIST
226	E2	STOP
227	E3	READ
228	E4	DATA
229	E5	RESTORE
230	E6	NEW
231	E7	BORDER
232	E8	CONTINUE
233	E9	DIM
234	EA	REM
235	EB	FOR
236	EC	GO TO
237	ED	GO SUB
238	EE	INPUT
239	EF	LOAD
240	FO	LIST
241	F1	LET
242	F2	PAUSE
243	F3	NEXT
244	F4	POKE
245	F5	PRINT
246	F6	PLOT
247	F7	RUN
248	F8	SAVE
249	F9	RANDOMIZE

десятичный код	шестнадцатиричный код	Символ
250	FA	IF
251	FB	CLS
252	FC	DRAW
253	FD	CLEAR
254	FE	RETURN
255	FF	COPY

ПРИЛОЖЕНИЕ 2.

Сообщение интерпретатора Бейсика.

Сообщения появляются в системном окне экрана при остановке выполнения программы. Сообщение имеет код, короткое сообщение, объясняющее, что произошло, номер строки и номер оператора внутри строки, где произошел останов.

Код сообщения	Сообщение	Содержание сообщения
0	OK	Успешное выполнение программы или команды
1	NEXT without FOR	NEXT без FOR
2	Variable not found	Переменная не найдена
3	Subscript wrong	Неверный индекс
4	Out of memory	Вне памяти
5	Out of screen	Вне экрана
6	Number too big	Слишком большое число
7	RETURN without GOSUB	RETURN без GOSUB
8	End of file	Конец файла
9	STOP statement	Останов по оператору STOP
A	Invalid argument	Ошибочный аргумент
B	Integer out of range	Целое вне диапазона
C	Nonsense in BASIC	Бессмыслица в Бейсике
D	BREAK — CONT repeats	Выполнение программы прервано командой BREAK — продолжение по команде CONTINUE
E	Out of DATA	Вне данных

Код сообщения	Сообщение	Содержание сообщения
F	Invalid file name	Неверное имя файла
G	No room for line	Нет места для строки
H	STOP in INPUT	Стоп при вводе
I	FOR without NEXT	FOR без NEXT
J	Invalid I/O device	Устройство ввода-вывода указано неверно
K	Invalid colour	Неверный цвет
L	BREAK into program	Выполнение программы прервано командой BREAK
M	RAMTOP no good	Адрес RAMTOP задан неправильно
N	Statement lost	Отсутствует оператор
O	Invalid stream	Ошибочный поток данных
P	FN without DEF	Функция пользователя не определена
Q	Parameter error	Ошибка в параметре функции
R	Tape loading error	Ошибка загрузки с ленты

ПРИЛОЖЕНИЕ 3.

Примеры программ.

Программа "русские буквы" определяет в области графики пользователя начертания русских букв, не совпадающих с латинскими.

```
10 REM Russean letters
```

```
20 FOR i=0 TO 21*8-1
```

```
30 READ code
```

```
40 POKE USR "a"+i, code
```

```
50 NEXT i
```

```
60 DATA 0, 62, 66, 66, 62, 34, 66, 0: REM A
```

```
70 DATA 0, 126, 64, 124, 66, 66, 124, 0: REM B
```

```
80 DATA 0, 68, 68, 68, 68, 68, 126, 2: REM C
```

```
90 DATA 0, 28, 36, 36, 36, 36, 126, 66: REM D
```

```
100 DATA 0, 60, 66, 30, 2, 66, 60, 0: REM E
```

```
110 DATA 0, 127, 73, 73, 73, 127, 8, 0: REM F
```

```
120 DATA 0, 126, 64, 64, 64, 64, 0: REM G
```

```
130 DATA 0, 66, 66, 66, 126, 2, 2, 0: REM H
```

```
140 DATA 0, 66, 70, 74, 82, 98, 66, 0: REM I
```

```
150 DATA 0, 65, 73, 62, 73, 73, 65, 0: REM J
```

```
160 DATA 34, 0, 126, 64, 120, 64, 126, 0: REM K
```

```
170 DATA 0, 30, 34, 34, 34, 34, 98, 0: REM L
```



```

180 DATA 0, 65, 73, 73, 73, 73, 127, 1: REM M
190 DATA 0, 65, 73, 73, 73, 73, 127, 0: REM N
200 DATA 0, 66, 66, 36, 24, 16, 96, 0: REM O
210 DATA 0, 126, 66, 66, 66, 66, 66, 0: REM P
220 DATA 0, 66, 66, 114, 74, 74, 114, 0: REM Q
230 DATA 0, 64, 64, 124, 66, 66, 124, 0: REM R
240 DATA 24, 66, 70, 74, 82, 98, 66, 0: REM S
250 DATA 0, 192, 64, 124, 66, 66, 124, 0: REM T
260 DATA 0, 76, 82, 114, 82, 82, 76, 0: REM U
270 FOR i=0 TO 20
280 PRINT "*****"; CHR$ (65+i); "*****"; "="; "*****"; CHR$ (144+i); "*****";
290 NEXT i

```

RUN

"A" = "Я"	"B" = "Б"
"C" = "Ц"	"D" = "Д"
"E" = "Э"	"F" = "Ф"
"G" = "Г"	"H" = "Ч"
"I" = "И"	"J" = "Ж"
"K" = "Е"	"L" = "Л"
"M" = "Ш"	"N" = "Ш"
"O" = "У"	"P" = "П"
"Q" = "Ы"	"R" = "Ь"
"S" = "Й"	"T" = "Ъ"
"U" = "Ю"	

Чтобы многократно пользоваться определенными в программе символами, необходимо измененную область определяемой графики выгрузить на ленту командой:

SAVE "RUS LET" CODE 65368, 168

Программа "Флаг" рисует на экране трехцветный национальный русский флаг.

```

10 REM National Russian flag
20 BRIGHT I: PAPER 5: CLS
30 PLOT 55, 15: DRAW 0, 144
40 LET i=I: LET p=1: INK 1
50 LET Y=87: GO SUB 100: REM DRAW BLUE PART
60 LET i=0: LET p=8: LET Y=56: INK 2: GO SUB 100: REM DRAW
RED PART
70 INK 7: LET Y=120: GO SUB 100: REM DRAW WHIGHT PART

```



```

80 INK 0: PAPER 7: STOP
100 FOR Y=Y TO Y+32
110 INVERSE i
120 PLOT PAPER p, 56, Y: DRAW PAPER p, 72, 0, -1: DRAW PAPER p,
71, 0, 1
130 NEXT Y
140 RETURN

```

В следующем примере приведена простая игровая программа
"bat and ball".

```

10 REM *** bat and ball ***
20 PRINT "key Z moves bat left"
30 PRINT "key M moves bat right"
40 REM set up ball
50 FOR i=USR "a" TO USR "a"+7
60 READ n: POKE i, n: NEXT i
70 DATA 60, 126, 255, 255, 255, 255, 126, 60
80 REM START the Game
85 PRINT "Press any key to start"
90 IF INKEY$="" THEN GO TO 90
100 PAPER 4: INK0: BORDER 1: CLS
110 LET b=14: LET r=10: LET l=10: LET dr=-1: LET dl=1
120 PRINT AT 20, b; " ■■■■ "
130 PRINT AT r-dr, l-dl; " "; AT r, l; CHR$ 144
140 IF INKEY$="Z" AND b<>0 THEN LET b=b-1: PRINT AT 20,
b; " ■■■■ "
150 IF INKEY$="M" AND b<>28 THEN PRINT AT 20,
b; " ■■■■ " : LET b=b+1
155 IF INKEY$="" THEN PRINT AT 20, b; " ■■■■ "
160 IF r+dr<0 THEN LET dr=-dr
170 IF l+dl<0 OR l+dl=32 THEN LET dl=-dl
180 IF r+dr<>20 THEN LET r=r+dr: LET l=l+dl: GO TO 130
190 IF l>=b AND l<b+4 THEN LET dr=-dr: BEEP. 005, 36: GO TO
130
200 FOR i=1 TO 2: LET r=r+dr: LET l=l+dl: PRINT AT r-dr, l-dl; " ";
AT r, l; CHR$ 144
205 NEXT i: BEEP 1, 0
210 INPUT "GAME OVER next game Y/N ?"; a$
220 IF a$="Y" THEN RUN
230 STOP

```


Лев Григорьевич РОЗИН
Сергей Юрьевич НИКОЛАЕВ

БЕЙСИК

Справочное пособие по программированию на
алгоритмическом языке для бытовых
микрокомпьютеров класса ZX SPECTRUM.

Ответственный редактор Л. РОЗИН.

Редактор Ю. ОСТРОУХОВ.

Технический редактор Г. ГРАМОТЕНКО.

Сдано в набор 21.05.92. Подписано в печать 29.07.92.

Формат 60×84/16. Бумага типографск. Гарнитура Таймс.

Печать офсетная. Усл. п. л. 5,03. Тираж 10000.

Заказ 1549.

Хозрасчетное производственное малое творческое предприятие

"Орфей". 344007, г. Ростов-на-Дону, пр. Ворошиловский, 18.

ЛР № 030088 от 29.08.91. Мининформпечати РФ.

РФЦП. 344008, г. Ростов-на-Дону, ул. Б. Садовая, 18.

