

**СИСТЕМА
ПОДГОТОВКИ ПРОГРАММ
15ИПГ 16**

БЭЙСИК-ИНТЕРПРЕТАТОР

(Вариант 3)

Руководство программиста

И5М1.419.001 Д23



СОДЕРЖАНИЕ

1. Введение	2
2. Описание языка	3
2.1. Общие сведения	3
2.2. Формы записи чисел	5
2.3. Переменные и массивы	5
2.4. Стандартные функции и функции пользователя	6
2.5. Выражения	8
2.6. Изменение значений переменных	9
2.7. Выходная печать	11
2.8. Условные и безусловные переходы	13
2.9. Циклы	14
2.10. Подпрограммы	16
2.11. Прекращение выполнения программы	16
2.12. Использование НМЛ	16
2.13. Использование перфоленты	20
2.14. Вспомогательные операторы	21
2.15. Программы в машинных кодах	22
3. Работа с интерпретатором	23
3.1. Загрузка в запуск	23
3.2. Режимы работы интерпретатора	24
3.3. Набор, редактирование и запуск программы	25
3.4. Остановы программы	26
3.5. Сообщения об ошибках	26
3.6. Рекомендации по отладке программ	28
3.7. Остановы интерпретатора	28
4. Дополнительные сведения	28
4.1. Распределение памяти	28
4.2. Общие принципы работы интерпретатора. Регистры	30
4.3. Представление исходной программы	30
4.4. Данные и служебные записи	34
4.5. Внутренние подпрограммы	35
4.6. Простейшие возможности изменения интерпретатора	38
5. Примеры программ	38
5.1. Корни квадратного уравнения	38
5.2. Биоритмы	38
5.3. Обращение матрицы	45
5.4. Информационно-справочная система	45
5.5. Посадка на Луну	53
5.6. Ввод-вывод на УСО	53
5.7. Организация ввода-вывода текстовой информации	61
5.8. Использование расширенной памяти объемом 128 Кбайт	62
6. Использование бэйсик-программ микро-ЭВМ «Электроника-60»	65
7. Справочное приложение	66
7.1. Общие формы записи операторов	66
7.2. Справочные таблицы	73
8. Литература	73
Лист регистрации изменений	74

1. ВВЕДЕНИЕ

1.1. В настоящем руководстве содержится описание языка Бэйсик и инструкция по работе с Бэйсик-интерпретатором (вариант 3) – в дальнейшем интерпретатор – системы подготовки программ 15ИПГ 16.

1.2. Бэйсик предназначен для решения математических и инженерных задач в режиме диалога человек – ЭВМ. Он позволяет программировать большой круг задач, сочетая в себе простоту и лёгкость для изучения и понимания с достаточными для многих применений возможностями.

1.3. Исходная программа и данные могут быть введены с клавиатуры пишущей машины, магнитной ленты или перфоленты. Результаты работы программы можно вывести на печать, перфоленту или магнитную ленту.

1.4. Интерпретатор рассчитан на работу со следующим минимальным комплектом системы:

а) устройство специализированное управляющее вычислительное «Электроника ДЗ-28» (в дальнейшем ДЗ-28) любого исполнения;

б) пишущая машина «Консул 260.1», подключенная к разъёму ПЕЧАТЬ.

Дополнительно при использовании восьмидорожечной перфоленты должны быть подключены непосредственно к ДЗ-28 или через контроллеры перфоратор и фотосчитыватель с кодами УПРАВЛЕНИЕ:

1200 – считыватель;

1500 – перфоратор.

Примечание. Интерпретатор предназначен для работы на ДЗ-28, выпущенных после ноября 1979 года.

1.5. В тексте использованы следующие сокращения:

МЛ – магнитная лента;

НС – номер строки;

ПМ – пишущая машина;

ПС – символ и клавиша ПМ «Перевод строки»;

ВК – символ и клавиша ПМ «Возврат каретки»;

ПФ – символ «Перевод формата» (код 0012).

1.6. Для более полного использования возможностей интерпретатора и понимания его реакций целесообразно ознакомиться с описанием программы И5М1.419.001 Д25 и её текстом И5М1.419.001 Д24.

1.7. Ниже приведён русский перевод имён операторов и служебных слов Бэйсика:

CALL – вызывать

CLEAR – очищать

COM (*common*) – общий

DATA – данные

DEF (*define*) – определять
DIM (*dimension – размерность*) – задание размеров массивов
END – конец
FOR – для
FN (*function*) – функция
GOSUB (*go subroutine*) – идти к подпрограмме
GOTO – идти к
IF – если
INPUT – ввести
LET – пусть
LIST – распечатать
LOAD – загрузить
NEXT – следующий
ON – по
OPEN – открыть
PRINT – напечатать
READ – прочитать
REM (*remark*) – комментарий
RESTORE – восстановить
RETURN – возврат
REWIND – перемотать
SAVE – хранить
SKIP – перепрыгнуть
STEP – шаг
STOP – останов
TAB (*tabulate*) – табулировать, располагать данные в виде таблиц
THEN – тогда
TO – к
RUN – пуск

2. ОПИСАНИЕ ЯЗЫКА

2.1. Общие сведения

2.1.1. Бэйсик-программа состоит из последовательности пронумерованных строк. В качестве разделителя строк используется символ ПС (или ВК).

В качестве нóмера строки может быть использовано любое целое положительное число от 1 до 7999. Номер строки может быть использован для ссылки на строку, например, при передачах управления в программе и, кроме того, служит для упорядочения строк в исходной программе при её вводе.

Максимальная длина строки – 100 символов.

Рекомендуется нумеровать соседние строки так, чтобы их номера отличались друг от друга на 5 или 10, что позволит при необходимости вставить между ними дополнительные строки.

2.1.2. Строка программы может состоять из одного или нескольких предложений, которые отделяются друг от друга символом «:». Предложение может начинаться только с оператора и должно заканчиваться в той же строке, где и началось.

В состав предложения входит оператор со всеми необходимыми для него служебными словами и параметрами. Оператором является слово, определяющее характер выполняемого действия.

Полный список операторов приведён в [разделе 7](#).

2.1.3. В произвольных местах строки в пределах её допустимой длины могут быть записаны пробелы. Все пробелы, кроме входящих в заключённый в апострофы текст, интерпретатором игнорируются и могут использоваться для облегчения чтения текста программы.

Однако слова Бэйсика должны быть записаны без пробелов, например, **GOTO**, **GOSUB** и т.п.

Пробелы в тексте, заключённом в апострофы, выводятся на печать наравне с другими символами.

2.1.4. В алфавит Бэйсика входят следующие символы:

- а) 26 заглавных латинских букв;
- б) 10 десятичных цифр от 0 до 9;
- в) 5 знаков препинания – точка, точка с запятой, двоеточие, запятая, апостроф;
- г) 5 знаков арифметических операций:
 - + знак сложения, знак плюс;
 - знак вычитания, знак минус;
 - * знак умножения;
 - / знак деления;
 - ^ знак возведения в степень);
- д) знаки операций отношения:
 - > больше;
 - < меньше;
 - = равно;
- е) круглые скобки:
 - (левая;
 -) правая;
- ж) пробел;
- з) знак #.

В заключённом в апострофы тексте можно использовать любые символы, соответствующие клавишам ПМ, кроме двоеточия и символов заобоя.

2.1.5. В любом месте программы можно записывать строки-комментарии, начинающиеся с оператора **REM**. Все символы строк, следующие за символом **REM**, ин-

терпретатором игнорируются. Текст строки не должен содержать символов заобоя и апострофов.

2.2. Формы записи чисел

2.2.1. В отличие от других алгоритмических языков, в Бэйсике определён лишь один тип чисел (вещественный). Форма записи числа близка к естественной. Для отделения целой части числа от дробной используется точка. Для записи очень больших или очень маленьких чисел применяется форма с десятичным множителем, порядок которого задаётся после латинской буквы **E**. Знак плюс и незначащие нули в записи числа (порядка) могут быть опущены.

Примеры:

Число	Запись на Бейсике
3,457	3.457 или .3457E1 или 345.7E-2 и т.д.
$-0,5 \cdot 10^6$	-0.5E6 или -5E+5 и т.д.
-0,00068	-6.8E-4 или -.68E-3 или -.00068 и т.д.
5	5 или +5 и т.д.

2.2.2. Ограничения при записи чисел:

- число, стоящее после буквы **E**, должно быть целым;
- количество цифр в целой или дробной части числа, а также в порядке, не должно превышать 12;
- модуль числа не должен превышать $(1 - 10^{12}) \cdot 10^{99}$.

2.2.3. Константа $\pi = 3.14159365159$ может быть задана специальным образом:

#PI.

2.3. Переменные и массивы

2.3.1. Величины, значения которых в процессе выполнения программы могут меняться, называются переменными. Для обозначения переменных в Бэйсике используются имена, состоящие либо из латинской буквы, либо из латинской буквы с последующей за ней цифрой, например, **A**, **B5**, **C1**, **D0**, **Z9**.

2.3.2. Наряду с одиночными переменными в Бэйсике можно использовать переменные-массивы (одномерные и двумерные).

Каждый массив определяется его именем, а также размерностью. Имя массива задаётся так же, как и имя простой (одиночной) переменной. Элементами массива являются индексированные переменные. Индексированная переменная обозначается именем массива и одним или двумя индексами, разделёнными запятой, в круглых скобках. Для одномерного массива индекс определяет порядковый номер элемента, для двумерного первый индекс – номер строки, второй – номер столбца, на пересечении которых расположен элемент.

Элементы одномерного массива, а также строки и столбцы двумерного массива нумеруются с нуля. Интерпретатор не отличает нулевые элементы массивов от одноименных простых переменных. Примеры индексированных переменных:

$A(0, 0) = A$ – нулевой элемент двумерного массива A ;

$M5(K, L)$ – элемент, расположенный на пересечении K -ой строки и L -го столбца;

$A9(1)$ – первый элемент одномерного массива $A9$.

2.3.3. Перед первым использованием индексированной переменной или массива целиком он должен быть описан одним из операторов **DIM** или **COM**. Описание массива сводится к объявлению его имени с указанием в скобках максимальных значений индексов.

Одним оператором, используя запятую, можно описать несколько массивов.

Отличие операторов **DIM** и **COM** друг от друга состоит в том, что области **DIM** и **COM** стираются разными командами: **CLEAR** и **CLEARC** соответственно (см. подраздел 2.14).

Примеры описаний массивов:

DIM A5(6, 8) – задаёт в области **DIM** двумерный массив $A5$ с семью строками и девятью столбцами;

COM A(0), C(255) – задаёт в области **COM** простую переменную A и одномерный массив C из 256 элементов.

Всем элементам описанных операторами **DIM** и **COM** массивов присваиваются нулевые значения.

2.3.4. Индексы в индексированных переменных и операторах **DIM** и **COM** могут задаваться с помощью индексных выражений. Значение этих выражений должно быть нулём или положительным числом, меньшим 256. За значение индекса в этом случае принимается целая часть (без округления) значения индексного выражения.

Например, **DIM R(I*J, K)** при значениях переменных к моменту выполнения оператора $I = 3.8$, $J = 3$, $K = 7.9$ определяет массив R с размерностью 11×7 (12×8 элементов).

2.3.5. Повторное определение массивов, а также определение массива, имеющего имя уже определённой простой переменной, не допускается без предшествующей очистки соответствующей области памяти. Недопустимо также использование оператора **COM** в теле цикла **FOR-NEXT**.

2.4. Стандартные функции и функции пользователя

2.4.1. В интерпретаторе определены стандартные функции, перечисленные в табл. 1. Функция задаётся трёхбуквенным именем и аргументом, заключённым в круглые скобки. Аргументом функции может быть произвольное арифметическое выражение. В табл. 1 выражение обозначено буквой X .

2.4.2. Кроме стандартных функций, имеется возможность использования в программе заранее определённых функций пользователя. Функция пользователя должна иметь имя из трёх латинских букв, первые две из которых – **FN**.

Аргумент функции, который может быть произвольным арифметическим выражением, заключается в круглые скобки.

Таблица 1

Обозначение функции	Значение функции	Примечание
DEG(X)	$\frac{180}{\pi} \cdot x$	перевод в градусы
RAD(X)	$\frac{\pi}{180} \cdot x$	перевод в радианы
SIN(X)	$\sin x$	аргумент в радианах
COS(X)	$\cos x$	то же
TAN(X)	$\operatorname{tg} x$	– " –
ASN(X)	$\arcsin x$	функция в радианах
ACS(X)	$\arccos x$	то же
ATN(X)	$\operatorname{arctg} x$	– " –
HSN(X)	$\operatorname{sh} x$	
HCS(X)	$\operatorname{ch} x$	
HTN(X)	$\operatorname{th} x$	
AHS(X)	$\operatorname{arsh} x$	
AHC(X)	$\operatorname{arch} x$	
AHT(X)	$\operatorname{arth} x$	
ABS(X)	$ x $	
EXP(X)	e^x	
EXT(X)	10^x	
LOG(X)	$\ln x$	
LGT(X)	$\lg x$	
SQR(X)	\sqrt{x}	
INT(X)	$\operatorname{int} x$	наибольшее целое, не превосходящее x
SGN(X)	$\operatorname{sgn} x = \begin{cases} 1, & \text{при } x > 0 \\ 0, & \text{при } x = 0 \\ -1, & \text{при } x < 0 \end{cases}$	
RND(X)	псевдослучайное число с равномерным распределением на $[0, 1]$	при $x = 0$ устанавливает начальное значение, при $x \neq 0$ – следующее

Функция пользователя определяется оператором

$$\text{DEF FN}\alpha(\beta)=\gamma,$$

где: $\text{FN}\alpha$ – имя функции (α – латинская буква);

β – имя простой переменной – формального параметра;

γ – произвольное арифметическое выражение.

Функция должна быть определена до её использования.

Недопустимо определение функции в теле цикла **FOR–NEXT**.

Например, оператор

```
DEF FNB(y)=y-3+y-2+5
```

определяет функцию **FNB** от формального параметра **y**, соответствующую выражению $y^3 + y^2 + 5$.

В следующих за ним операторах программы по аналогии со стандартными функциями может быть использована функция **FNB(x)**, процедура вычисления значения которой определена в операторе **DEF**. В программе может иметься переменная с именем формального параметра, которая, однако, не будет иметь с ним ничего общего.

Арифметическое выражение, стоящее справа от знака равенства в операторе **DEF**, может содержать любые определённые к моменту вычисления функции пользователя переменные и функции, кроме самой определяемой функции. Это выражение может содержать или не содержать формальный параметр.

Например, **DEF FNA(Z)=SIN(X)+COS(X)** определяет функцию по имени **FNA**, не зависящую от формального параметра **Z**, т.е. не зависящую от значения аргумента при вызове функции:

$$FNA(5)=FNA(SIN(X)+COS(X))=\sin x + \cos x$$

Если же функция определена в виде

$$DEF FNA(Z)=SIN(Z)+COS(X), \text{ то } FNA(5)=\sin 5 + \cos x, \text{ а}$$

$$FNA(COS(X)+EXP(X))=\sin(\cos x + e^x) + \cos x.$$

Таким образом, для вычисления значения функции **FNA** значение её аргумента подставляется вместо формального параметра в определяющее функцию выражение.

2.4.3. Стандартные функции и функции пользователя допускают рекурсивное использование, например:

$$SIN(SIN(X))=\sin(\sin(x));$$

FNA(2+2*FNA(2))=100 для функции, предварительно определённой оператором **DEF FNA(X)=X*X**.

2.5. Выражения

2.5.1. Арифметическое выражение состоит из констант, переменных и функций, соединённых знаками арифметических операций:

\uparrow – знак возведения в степень;

$*$ – знак умножения;

$/$ – знак деления;

$+$ – знак сложения;

$-$ – знак вычитания.

2.5.2. При записи для указания порядка выполнения операций используются круглые скобки. Внутри скобок операции выполняются слева направо в соответствии с общепринятым приоритетом:

- а) вычисление значений функций;
- б) возведение в степень;
- в) умножение и деление;
- г) сложение и вычитание.

В сомнительных для программиста и пользователей программы случаях типа $x \rightarrow y \rightarrow z$ порядок действий рекомендуется указывать явно: $(x \rightarrow y) \rightarrow z$ или $x \rightarrow (y \rightarrow z)$.

Примеры:

Формула	Выражение Бэйсика
$\frac{ax^2 + bx + c}{d - 2,5}$	$(A*X^2+B*X+C)/(D-2.5)$
$\frac{2 \sin x \cos x}{\sqrt{b}}$	$2*SIN(X)*COS(X)/SQR(B)$

2.5.3. Все знаки операций в выражении должны быть записаны явно.

Два знака операции подряд недопустимы. Например, не допускается запись $2A+B$ вместо $2*A+B$ или $A*-B*C$ вместо $A*(-B)*C$.

2.6. Изменение значений переменных

2.6.1. Простейшим оператором изменения значения переменной является оператор присваивания **LET**:

LET $\alpha = \beta$,

где: α – простая или индексированная переменная;

β – произвольное арифметическое выражение.

Примеры:

LET A1=5 присваивает переменной **A1** значение 5;

LET B5(7,8)=A1 присваивает индексированной переменной **B5(7,8)** значение переменной **A1**;

LET X=X*SIN(X) присваивает переменной **X** новое значение, равное функции $x \sin x$ старого значения.

Примечания:

1. Оператор **LET** выполняет функции определения и записи в область **DIM** новых простых переменных программы. Оператор **LET X=0** для новой переменной **X** эквивалентен операторам **DIM X(0)** или **DIM X(0,0)**.

2. При записи и вводе текста программы оператор **LET** может быть пропущен. Однако он вставляется интерпретатором в соответствующие места программы и выдаётся на носители при выводе текста.

2.6.2. Второй приём для изменения значений переменных базируется на использовании блока данных.

Блок данных задаётся в исходной программе одним или несколькими операторами **DATA**, стоящими в произвольном месте программы.

Например, оператор **20 DATA -1,.75,.2E-6**, если он единственный в программе, задаёт блок данных из трёх чисел: -1 ; $0,75$; $0,2 \cdot 10^{-6}$, а операторы в программе

```
10 DATA 1
  :
120 DATA 1,2
  :
500 DATA 3,4,5
```

– блок данных из шести чисел: 1, 1, 2, 3, 4, 5.

Предложение с оператором **DATA** должно быть единственным или последним предложением в строке.

2.6.3. Интерпретатор допускает использование в операторах не только чисел, но и произвольных выражений. Естественно, использованные в выражениях переменные должны быть определены до чтения соответствующего элемента блока данных.

2.6.4. Чтение из блока данных и присваивание переменным значения производится оператором **READ**.

Чтение производится по внутреннему указателю блока данных. При запуске программы указатель устанавливается в исходное положение. После выборки очередного значения указатель блока данных перемещается к следующей позиции. Попытка чтения за пределами блока данных является ошибкой.

Одним оператором **READ** могут быть присвоены значения нескольким переменным. В теле оператора эти переменные разделяются запятыми.

Пример:

```
10 DATA 1,2,3,4
20 DATA 6,7
  :
40 READ X(5,8)
50 READ Y,Z(4)
50 READ A1,A2,A3
```

Переменным **X(5,8)**, **Y**, **Z(4)**, **A1**, **A2**, **A3** будут присвоены, соответственно, значения 1, 2, 3, 4, 6, 7.

При необходимости пропуска некоторых элементов блока данных можно использовать оператор типа **READ X,X,X**.

2.6.5. Восстановление указателя блока данных в исходное положение для повторного чтения производится оператором **RESTORE**:

```
10 DATA 1,2,3,4
  :
50 READ X,Y,Z
  :
100 RESTORE
110 READ Z,Z,Z
```

В строках 110 и 50 переменной **Z** присваивается значение третьего элемента блока данных (3).

2.6.6. Описанные выше операторы **READ** и **LET** присваивают переменным значения, заданные непосредственно в тексте программы.

Для присваивания переменным значений, вводимых во время работы программы по её запросу с клавиатуры пишущей машины, служит оператор **INPUT** со списком переменных. В оператор может быть включён произвольный текст, заключённый в апострофы. Этот текст будет выдан на печать при выполнении оператора **INPUT**. При отсутствии в теле оператора текста печатается вопросительный знак. Работа программы приостанавливается для ввода оператором необходимых значений.

Оператором могут быть введены как числа в любой допустимой форме, так и любые выражения, содержащие определённые к данному моменту переменные.

Вводимые значения присваиваются последовательно переменным из списка оператора **INPUT**. Разделителем значений при вводе является запятая, ввод должен быть завершён клавишей **ВК (ПС)** или клавишей «\». В последнем случае строка печати переведена не будет.

Примеры:

а) оператор **INPUT 'ВВЕДИТЕ K,N(10)' K,N(10)** печатает заключённый в кавычки текст и принимает с клавиатуры ПМ значения переменных **K** и **N(10)**;

б) оператор **INPUT J** печатает в новой строке вопросительный знак и принимает значение переменной **J**.

Примечание. Так же как и оператор **LET**, оператор **INPUT** определяет новые переменные программы, записывая их в область **DIM**.

2.6.7. Ввод данных с перфоленты и машинной ленты с присваиванием значений переменным и массивам может быть произведён оператором **DATA LOAD**, описанном в последующих подразделах 2.12 и 2.13.

2.7. Выходная печать

2.7.1. Печать результатов вычислений, пояснительных текстов и графиков осуществляется с помощью оператора **PRINT**. Тело оператора **PRINT** носит название выводного списка. Элементами этого списка могут быть числа, переменные, выражения, заключённые в кавычки тексты, а также служебные слова и спецификации формата.

Разделителями списка, служащими одновременно для управления печатью, являются запятая и точка с запятой.

2.7.2. В интерпретаторе реализовано три вида форматов печати числа. Формат печати задаётся в списке оператора **PRINT** спецификацией, расположенной между восклицательными знаками (!). Отрицательные числа выводятся на печать со знаком. Знак **+** на печать не выводится, заменяется пробелом. Перед печатью числа производится его округление с точностью до последней печатаемой цифры.

2.7.3. Спецификация формата **!Fn₁.n₂!**, где n_1 и n_2 – цифры, означает, что на печать выводится знак числа, n_1 цифр до десятичной точки, десятичная точка, n_2 цифр после десятичной точки, буква **E**, знак порядка и две цифры порядка.

Незначащие нули и буква **E** при нулевом порядке заменяются пробелами.

Любое число в формате **!Fn₁.n₂!** занимает $n_1 + n_2 + 5$ позиций.

2.7.4. Спецификация формата **!n₁.n₂!**, где n_1 и n_2 – цифры, означает, что на печать выводится знак числа, n_1 цифр до десятичной точки, точка и n_2 цифр после точки.

Незначащие нули в целой части числа заменяются пробелами. Если число не может быть напечатано в заданном формате, выводится столько символов, сколько позиций было отведено на печать числа, т.е. $n_1 + n_2 + 2$.

2.7.5. Спецификация формата **!E!** означает, что на печать число выводится в форме с плавающей запятой: знак числа, точка, 12 цифр мантииссы, символ **E**, знак порядка и две цифры порядка. Если порядок числа равен 0, то порядок и символ **E** не печатаются, вместо них выводятся пробелы.

2.7.6. При запуске интерпретатора задаётся формат печати **!F1.9!**. Указание о формате действует до замены его новым указанием.

2.7.7. Если в списке оператора **PRINT** есть последовательность символов, заключённая в кавычки, она выдаётся на печать без изменения.

2.7.8. Слово **OPEN** в списке оператора **PRINT** выводит на печать имя последнего считанного с МЛ файла.

2.7.9. Если в списке оператора **PRINT** есть слово **TAB**, то происходит установка каретки ПМ на позицию, координата которой равна целой части значения выражения, записанного после **TAB**. Допустимый диапазон значений выражения [0, 101].

2.7.10. Вся строка печати, длиной 100 позиций, условно разделяется на 5 зон по 20 позиций в каждой зоне. Зонная печать числа обеспечивается использованием запятой в качестве разделителя между элементами списка. Каждая запятая вызывает перемещение каретки к началу следующей зоны.

2.7.11. Для более плотной печати в качестве разделителя используется точка с запятой, вызывающая перемещение каретки ПМ вправо на одну позицию.

2.7.12. Если в текущей строке для печати очередного числа не хватает места, то оно начинает печататься с начала следующей строки.

При выводе текста, выходящего за последнюю позицию печати (101) автоматически выдаётся команда возврата каретки.

2.7.13. Числа, переменные и выражения при последовательной их записи в списке оператора **PRINT** должны обязательно записываться с разделителями – точкой с запятой или запятой. При нежелательности связанных с этими разделителями перемещений каретки можно в качестве разделителя использовать спецификацию формата.

Тексты, заключённые в кавычки могут быть записаны подряд, без разделителей.

2.7.14. После вывода на печать элементов списка оператора **PRINT**, как правило, на ПМ выдаётся команда возврата каретки. Возврат каретки подавляется, если:

а) список завершён разделителями запятая или точка с запятой, а также спецификацией формата;

б) последним элементом списка является слово **TAB** с сопутствующим выражением.

Пустой (без списка) оператор **PRINT** также вызывает возврат каретки.

2.7.15. Примеры использования оператора **PRINT**:

а) в результате выполнения строки:

```
A=729: PRINT !3.0! 'КВАДРАТНЫЙ КОРЕНЬ ИЗ 'A' ='SQR(A)
```

будет напечатано:

```
КВАДРАТНЫЙ КОРЕНЬ ИЗ 729 = 27
```

б) результат выполнения строки **PRINT !F2.9!,#PI,PI/2 -**

```
31.415936536E-01      15.707963268E-01
```

в) оператор **PRINT** может использоваться для построения графиков. Например, после выполнения строки

```
FOR X=-6 TO 6: PRINT TAB (X-2)'*': NEXT X
```

будет получено на ПМ:

```

                                     *
                                 *
                             *
                         *
                    *
                *
            *
        *
    *
 *

```

2.8. Условные и безусловные переходы

2.8.1. Нумерация строк определяет естественный порядок выполнения программы. Для его изменения используются операторы условных и безусловных переходов.

2.8.2. Безусловный переход производится к началу строки, номер которой записал в теле оператора **GOTO**. Например, **GOTO 452** задаёт переход на строку 452.

При отсутствии строки с заданным в **GOTO** номером программа будет выполняться со строки со следующим (бóльшим) номером.

2.8.3. Простейшим оператором, позволяющим организовать в программе условные переходы, является оператор **IF-THEN**, простейшая форма которого:

```
IF  $\alpha$   $\otimes$   $\beta$  THEN HC,
```

где: α и β – произвольные арифметические выражения;

\otimes – символ или символы операции отношения:

- < – меньше;
- > – больше;
- = – равно;
- >< или <> – не равно;
- =< или <= – меньше или равно;
- => или >= – больше или равно;

НС – номер строки, к которой выполняется переход при выполнении условия $\alpha \otimes \beta$.

При невыполнении условия переход осуществляется к следующей по порядку строке.

Примечание. Слово **THEN** в приведённой выше форме оператора может быть заменено словом **GOTO** (оператор **IF–GOTO**).

2.8.4. Интерпретатор допускает запись в операторе **IF–THEN** вместо нóмера строки любого оператора Бэйсика, в том числе и другого оператора **IF**. В этом случае при выполнении условия управление передаётся оператору, записанному после слóва **THEN**, при невыполнении – как обычно, следующей строке.

2.8.5. Для организации ветвления программы по многим направлениям может быть полезен оператор **ON**. В теле оператора записывается произвольное арифметическое выражение, целая часть значения которого является номером строки перехода. Значение выражения должно лежать в интервале [1, 8000].

2.9. Циклы

2.9.1. Циклы любого вида могут быть организованы с помощью оператора **IF**. Для упрощения организации циклов с заданным числом повторений могут быть использованы специальные операторы **FOR** и **NEXT**. Цикл задаётся заголовком (оператором **FOR**), телом цикла, состоящим из предложений повторяющего участка программы и оператором **NEXT**, завершающим цикл.

2.9.2. Заголовок цикла записывается в виде

FOR $\alpha = \beta$ **TO** γ **STEP** δ ,

где: α – управляющая переменная;

β – выражение, задающее начальное значение переменной при входе в тело цикла;

γ – выражение, определяющее граничное значение переменной при выполнении цикла;

δ – выражение, определяющее приращение управляющей переменной при очередном выполнении тела цикла.

Завершающий цикл оператор **NEXT** записывается в виде **NEXT** α , где α – управляющая переменная цикла.

При организации цикла операторы **FOR** и **NEXT** должны иметь одну и ту же управляющую переменную.

2.9.3. При выполнении оператора **FOR** определяются параметры цикла: начальное значение, граничное значение и шаг приращения, управляющей переменной присваивается начальное значение. Если параметры цикла заданы так, что достичь конечного значения невозможно (конечное значение больше начального при отрицательном шаге приращений или конечное значение меньше начального при положительном шаге приращения), то цикл не выполняется и управление передаётся предложению, следующему за оператором **NEXT** этого цикла.

2.9.4. При выполнении оператора **NEXT** производится изменение значения управляющей переменной на величину шага приращения и проверяется выполнение условия выхода из цикла. Цикл прекращается, когда значение управляющей переменной оказывается строго больше (при положительном шаге приращения) или строго меньше (при отрицательном шаге приращений) граничного значения. Управление при этом передаётся предложению, следующему за оператором **NEXT**.

Если условие выхода из цикла не выполнено, управление передаётся предложению, следующему за оператором **FOR** этого цикла, т.е. в начало тела цикла.

2.9.5. Внутри одного цикла могут располагаться другие циклы. Внутренние циклы по отношению к внешнему называются вложенными. Интерпретатор допускает семь уровней вложения циклов. Внутренний и внешний циклы при вложении должны быть организованы по разным управляющим переменным.

2.9.6. Если в операторе **FOR** заголовка цикла отсутствует слово **STEP**, то интерпретатор устанавливает шаг приращения, равный +1. Задание нулевого шага приводит к сообщению об ошибке.

2.9.7. При выходе из цикла значение управляющей переменной равно последнему использованному в теле цикла значению. Если цикл не выполняется ни разу, за управляющей переменной сохраняется начальное значение.

2.9.8. Вход внутрь тела цикла, минуя оператор **FOR**, недопустим.

2.9.9. Пример правильной организации вложенных циклов:

```
10 FOR A=1 TO X STEP Y
  :
50 FOR C=-1 TO 10
  :
90 FOR D=K+1 TO K+10
  :
150 NEXT D
  :
180 NEXT C
  :
200 NEXT A
  :
```


2.10. Подпрограммы

2.10.1. Для организации подпрограмм используются операторы **GOSUB** и **RETURN**.

По оператору **GOSUB**, имеющему вид **GOSUB NC**, где **NC** – номер строки, управление передаётся строке, номер которой указан после имени оператора. При этом запоминается адрес обращения к подпрограмме.

2.10.2. Подпрограмма, которая начинается на строке с указанным номером, должна заканчиваться оператором **RETURN**. При выполнении оператора **RETURN** управление передаётся предложению, следующему за предложением обращения к подпрограмме.

2.10.3. Если в программе нет строки с указанным после **GOSUB** номером, управление передаётся строке с ближайшим бóльшим номером.

2.10.4. Внутри подпрограммы можно использовать переход на другую подпрограмму низшего уровня. Максимальное количество уровней подпрограмм равно 16.

2.11. Прекращение выполнения программы

2.11.1. Запланированный останов программы задаётся операторами **STOP** или **END**. Для интерпретатора эти операторы эквивалентны. Однако в целях обеспечения возможности последующей компиляции программы рекомендуется для останова внутри программы использовать оператор **STOP**, а последним оператором записывать **END**.

2.11.2. Программа автоматически останавливается также после выполнения строки с наибольшим номером.

2.11.3. Не предусмотренные в программе остановы, вызванные ошибочными ситуациями или вмешательством оператора, рассмотрены в [разделе 3](#).

2.12. Использование МЛ

2.12.1. Интерпретатор позволяет использовать магнитную ленту как для хранения данных, так и для хранения текстов программ или их сегментов.

2.12.2. Применяемые при работе с интерпретатором кассеты с магнитной лентой должны иметь прозрачный ракорд.

Рекомендуется проведение предварительного контроля и подготовки используемых кассет в соответствии с Инструкцией по эксплуатации И5М3.857.100 ИЭ.

2.12.3. Интерпретатор записывает программу и данные на МЛ элементарными записями длиной 256 байт с двукратным дублированием. На одну сторону кассеты может быть записано примерно 400 элементарных записей.

В зависимости от назначения, записи разделяются на типы. Необходимые для программиста сведения о типах записи приведены ниже.

2.12.4. При записи текста программы на МЛ интерпретатор формирует программный файл, состоящий из следующих записей:

- а) заголовок файла;

б) записи текста строк программы;

в) закрывающая запись файла.

В заголовок программного файла записывается его имя.

Для записи программного файла на МЛ используется оператор **SAVE** с указанием, в общем случае, начального и конечного номеров строк записываемой секции программы.

Непосредственно после имени оператора может быть записан заключённый в апострофы произвольный текст – имя программного файла.

Примеры:

а) оператор **SAVE 'СЕКМЕНТ 250' 250,267** записывает программный файл с именем **СЕКМЕНТ 250**, содержащий строки текущей программы с 250 по 267 включительно;

б) оператор **SAVE 1,100** записывает на МЛ безымянный программный файл, содержащих тексты строк 1-100.

Примечания: 1. Оператор **SAVE** без указания номеров строк записывает на МЛ всю программу, находящуюся в ОЗУ. Такой оператор эквивалентен оператору **SAVE 1,7999**. 2. При наличии в операторе **SAVE** одного номера строки на печать будет выдано сообщение об ошибке.

2.12.5. Оператор **SAVE END** записывает на МЛ служебную запись, являющуюся признаком конца ленты. Этот признак служит ограничителем при чтении программы или данных.

2.12.6. Чтение с МЛ и загрузка в ОЗУ программных файлов производится оператором **LOAD**:

LOAD 'α' HC1,HC2,

где: **α** – имя файла;

HC1 и **HC2** – номера строк.

При выполнении оператора **LOAD** интерпретатор выполняет следующие действия:

а) исключает из программы текст строк, имеющих номера от **HC1** до **HC2** включительно и запоминает начальный адрес загрузки;

б) начиная с текущего положения ленты, ищет заголовок программного файла с заданным именем;

в) загружает записи текста строк программы, начиная с запомненного адреса, раздвигая при необходимости текст находящейся в ОЗУ программы;

г) по прочтению закрывающей записи файла передаёт управление строке с номером, следующим за номером строки, содержавшей оператор **LOAD**.

Если файл прочитан с ошибками, то уже загруженная часть программы стирается и чтение продолжается с поиска заголовка программного файла.

При невозможности загрузки файла по записи «КОНЕЦ ЛЕНТЫ» выполнение оператора завершается.

Примечания:

1. Если имя файла в операторе не задано, производится загрузка очередного программного файла;
2. Сравнение имён заданного и читаемого файлов ведётся по шести первым символам имени (включая пробелы).
3. При отсутствии в записи оператора номеров строк **HC1** и **HC2** стирается вся программа.
4. Задание только одного номера строки является ошибкой.
5. Имя загруженного файла может быть напечатано оператором **PRINT**, содержащим в списке слово **OPEN**. При печати это имя дополняется слева буквой **P** и пробелом, указывающими на тип файла.

2.12.7. При записи данных интерпретатор формирует на МЛ блок данных, состоящий из следующих записей:

- а) начало блока;
- б) записи данных;
- в) конец блока.

Запись блока данных производится оператором **DATA SAVE**, содержащим список блока. Элементами списка могут быть произвольные арифметические выражения и массивы. Последние задаются именем массива со следующими за ним открывающей и закрывающей скобками. Элементы списка блока разделяются запятыми.

Например, оператор **DATA SAVE A()** записывает на МЛ блок данных, содержащий массив **A**, а оператор **DATA SAVE B(3), SIN(X+Y)*Z, B(), C** – третий элемент массива **B**, результат вычисления выражения $\sin(x + y) * z$, массив **B** и простую переменную **C**.

2.12.8. Блоки можно объединять в файлы данных. Файл организуется путём записи заголовка оператором **DATA SAVE OPEN**, содержащим (необязательное) имя файла, заключённое в кавычки, и конца файла оператором **DATA SAVE END**.

Например, файл данных «МАССИВЫ», состоящий из трёх блоков, содержащих соответственно массивы **A**, **B** и **C**, может быть сформирован следующим образом:

```
100 DATA SAVE OPEN 'МАССИВЫ'  
110 DATA SAVE A()  
120 DATA SAVE B()  
130 DATA SAVE C()  
140 DATA SAVE END  
250 SAVE END
```

Последний оператор записывает признак «КОНЕЦ ЛЕНТЫ».

При записи вместо строк 110-130 оператора **110 DATA SAVE A(), B(), C()** тот же файл состоял бы из одного блока данных вместо трёх.

2.12.9. Чтение блоков данных с МЛ и присваивание переменным прочитанных значений производится оператором **DATA LOAD**. В списке оператора, кроме простых и индексированных переменных, могут быть заданы целые массивы. Например,

DATA LOAD A, B(2,7), C() присваивает переменным **A**, **B(2,7)** и элементам массива **C** последовательные значения из загруженного с МЛ блока данных.

Чтение блока производится с текущего положения МЛ.

Выполнение оператора завершается в следующих случаях:

а) после чтения очередного блока присвоены значения всем элементам списка **DATA LOAD**;

б) прочитана запись «КОНЕЦ ФАЙЛА ДАННЫХ»;

в) прочитана запись «КОНЕЦ ЛЕНТЫ».

Таким образом, возможны случаи сохранения за всеми или частью элементов списка или массива прежних значений без выдачи сообщений оператору.

2.12.10. Вывод магнитной ленты на первый блок файла данных производится оператором **DATA LOAD α**, где **α** – имя файла данных. Сравнение имён производится по первым шести символам.

Если заголовок файла данных с нужным именем не найден, выполнение оператора завершается по прочтении записи «КОНЕЦ ЛЕНТЫ». Очередной заголовок может быть прочитан оператором **DATA LOAD OPEN**. Имя файла может быть напечатано оператором **PRINT** со служебным словом **OPEN**. При печати имя файла данных дополняется слева буквой **D** и пробелом.

2.12.11. Перемотка МЛ в начало производится оператором **REWIND**.

Программные файлы, отдельные блоки и файлы данных можно пропустить с помощью оператора **SKIP**.

Для пропуска заданного числа блоков данных после имени оператора записывается выражение, определяющее количество пропускаемых блоков. Оператор выполняется путём чтения записей с МЛ и подсчёта числа записей «КОНЕЦ БЛОКА».

Выход из оператора производится по чтению нужного числа записей «КОНЕЦ БЛОКА» или по записи «КОНЕЦ ЛЕНТЫ».

Пропуск заданного числа файлов задаётся записью после выражения, определяющего количество, буквы **F**. При его выполнении подсчитывается количество записей «КОНЕЦ ФАЙЛА». Программные файлы и файлы данных при подсчёте не различаются. Примеры:

SKIP 2 – пропуск двух блоков данных;

SKIP 3F – пропуск трёх файлов,

2.12.12. Анализ варианта завершения команд **LOAD**, **DATA LOAD**, **SKIP** можно производить операторами **IF END-THEN** и **IF END F-THEN**.

Если выполнение указанных команд завершилось по чтению записи «КОНЕЦ ЛЕНТЫ», после оператора **IF END** будет выполнен оператор, стоящий после слова **THEN** (или осуществлён переход к номеру строки). Оператор **IF END F-THEN** аналогично анализирует запись «КОНЕЦ ФАЙЛА ДАННЫХ».

В случае отсутствия в буфере МЛ указанных типов записи управление будет передано следующей строке.

Пример:

```

20 DATA LOAD A(),B()
30 IF END THEN 150
40 IF END F THEN 100
50 REM ВСЕ ЭЛЕМЕНТЫ МАССИВОВ ПОЛУЧИЛИ НОВЫЕ ЗНАЧЕНИЯ
:
100 REM НЕ ВСЕ ЭЛЕМЕНТЫ МАССИВОВ ПОЛУЧИЛИ НОВЫЕ ЗНАЧЕНИЯ,
110 REM ЧТЕНИЕ ЗАВЕРШЕНО ПО КОНЦУ ФАЙЛА ДАННЫХ
:
150 REM НЕ ВСЕ ЭЛЕМЕНТЫ МАССИВОВ ПОЛУЧИЛИ НОВЫЕ ЗНАЧЕНИЯ,
160 REM ЧТЕНИЕ ЗАВЕРШЕНО ПО КОНЦУ ЛЕНТЫ
:

```

2.12.13. В операторах **LOAD**, **SAVE**, **DATA LOAD**, **DATA SAVE** по аналогии с записью этих операторов применительно к работе с перфолентой (см. следующий подраздел) после имени оператора можно записывать «НОМЕР НМЛ» – **#0**.

2.13. Использование перфоленты

2.13.1. Так же как и на МЛ, на перфоленте можно хранить программу и данные.

Перфорация этих лент может быть выполнена интерпретатором или любым другим способом, например, на устройствах типа УПДЛ в кодах ГОСТ 13052-74.

Бит контроля на чётность при вводе интерпретатором игнорируется, при выводе – заменяется нулём.

2.13.2. Текст программы на перфоленте представляется в обычном символьном виде. Разделителями строк на вводимой перфоленте могут быть символы **ВК** или **ПС**. Интерпретатор при вводе игнорирует пустые пробивки и коды забоя. Программа на перфоленте должна быть завершена кодом 0012 (**ПФ**). Выводимый интерпретатором текст в качестве разделителя имеет символ **ПС**.

2.13.3. Данные на вводимой перфоленте должны состоять из чисел, разделённых запятыми. Число может быть записано в любом допустимом для Бэйсика виде.

Интерпретатор выводит данные на перфоленту в формате, соответствующем спецификации **!E!** оператора **PRINT**. Незначащие нули мантиссы не выводятся.

Блок данных на перфоленте должен быть завершён символом **ПФ** (код 0012).

2.13.4. Для работы с перфолентой используются те же операторы, что и для МЛ: **DATA SAVE**, **DATA LOAD**, **SAVE**, **LOAD** с обязательным указанием «НОМЕРА УСТРОЙСТВА», задаваемого символами **#1**.

Например, **LOAD #1** и **DATA SAVE #1 C()** относятся к перфоленте, а **LOAD [LOAD #0]** и **DATA SAVE C()** [или **DATA SAVE #0 C()**] – к магнитной ленте.

2.13.5. Вывод программы на перфоленту производится оператором **SAVE #1**. По оператору выводится вся находящаяся в ОЗУ программа.

2.13.6. Чтение программы или её сегментов с перфоленты производится оператором **LOAD #1**.

Предварительного стирания находящейся в памяти программы оператор не производит.

Реакция интерпретатора на принимаемые с перфоленты символы текста в точности совпадает с реакцией на те же символы, набираемые с клавиатуры ПМ (см. раздел 3).

По концу программы (символ ПФ) управление передаётся строке со следующим за номером строки оператора **LOAD** номером.

2.13.7. Вывод данных на перфоленту производится оператором **DATA SAVE #1** со списком выводимых элементов. Правила записи списка описаны в п. 2.12.7.

2.13.8. Ввод данных с перфоленты производится оператором **DATA LOAD #1** со списком вводимых элементов (см. п. 2.12.9).

Независимо от длины списка блок читается до конца. Если для всего списка в блоке данных недостаточно, значения оставшихся переменных и элементов массивов не изменяются.

2.14. Вспомогательные операторы

2.14.1. К этой группе отнесём операторы стирания программы и данных и вывода текста программы на печать.

2.14.2. В некоторых случаях в программах могут быть полезными операторы очистки памяти:

CLEAR C – очистка области **COM**;

CLEAR D – очистка области **DIM**;

CLEAR F – сброс указателя уровней циклов;

CLEAR S – сброс указателя уровней подпрограмм;

CLEAR HC₁, HC₂ удаление из программы строк от **HC₁** до **HC₂** включительно, перемотка МЛ в начало;

CLEARP HC₁, HC₂ – удаление из программы строк от **HC₁** до **HC₂** включительно.

Примечания:

1. Если в операторах **CLEAR** и **CLEARP** задан только один номер строки, будет стёрта только эта строка. Оператор **CLEARP** без номеров строк игнорируется, **CLEAR** – вызывает перемотку МЛ.

2. В область **COM** интерпретатор записывает массивы переменных, определённых оператором **COM**, и адреса определений функций пользователя (**DEF**). В область **DIM**, кроме массивов, определённых одноимённым оператором, записываются также простые переменные.

2.14.3. Вывод на печать текста программы (листинга) производится оператором **LIST** вида **LIST HC₁, HC₂**, где **HC₁** и **HC₂** – номерá строк. По этому оператору распечатывается часть программы, ограниченная строками с **HC₁** по **HC₂** (включительно).

При указании одного номера строки печатается текст только этой строки.

Если номерá строк не заданы, выводится весь текст программы, находящейся в памяти.

2.15. Программы в машинных кодах

2.15.1. В интерпретаторе реализована возможность включения в текст программы фрагментов в машинных кодах, а также обращения к подпрограммам в машинных кодах (внешним подпрограммам). Последние должны быть загружены в память во время начального диалога (см. [раздел 3](#)).

2.15.2. Включение в языковую программу фрагментов, написанных в машинных кодах, обеспечивается оператором **CMD** со списком кодов.

Список кодов команд должен располагаться в одной строке. Элементы списка могут разделяться запятой. Каждый элемент является шестнадцатеричным кодом команды в принятом для ДЗ-28 виде. Пробелы в списке, как обычно, игнорируются.

Обычно этот оператор используется для организации нестандартных операций ввода-вывода.

Например, оператор **CMD 1406 0207, 1406 0310** вызывает печать на пишущей машине символов «'» и «:», которые нельзя вывести обычным оператором **PRINT**.

2.15.3. Обращение к внешним подпрограммам, загруженным в процессе начального диалога, осуществляется оператором **CALL** с номером внешней подпрограммы. Номером подпрограммы может быть любое целое число от 1 до 7999.

При выполнении этой подпрограммы интерпретатор после чтения нóмера передаёт управление по адресу загрузки внешней подпрограммы в ОЗУ. В регистре **S3** при этом записан код символа, стоящего после нóмера внешней подпрограммы в предложении обращения, регистр **R1** содержит адрес следующего символа строки.

За номером внешней подпрограммы в операторе вызова **CALL** могут быть записаны параметры вызова. Правила записи и разделителя параметров определяются самой внешней подпрограммой.

Во внешних подпрограммах, как и в операторе **CMD**, допускается использование внутренних подпрограмм интерпретатора.

Внешняя подпрограмма должна завершаться командой **RTS** (0511). Перед этой командой в **S3** должен быть записан (или сохранён) код завершающего предложение **CALL** символа (двоеточие или **ПС**), а в **R1** – адрес следующего символа текста программы.

2.15.4. Для сохранения работоспособности интерпретатора при использовании операторов **CALL** и **CMD** необходимо заботиться о сохранности его самого, таблиц адресов подпрограмм и т.п. Дополнительные сведения, необходимые для эффективного применения программ в машинных кодах, приведены в [разделе 4](#).

2.15.5. Требования к форматам записи внешней программы на магнитной ленте изложены в [подразделе 3.1](#).

3. РАБОТА С ИНТЕРПРЕТАТОРОМ

3.1. Загрузка и запуск

3.1.1. Подключите к ДЗ-28 пишущую машину «Консул 260.1» и, при необходимости, перфоратор и фотосчитыватель в соответствии с Инструкцией по эксплуатации И5М3.857.100 ИЭ.

3.1.2. Включите в сеть всё используемое оборудование. После включения ДЗ-28 должно быть

$$(PгX) = (PгY) = 0; \quad AШ = НШ = 0.$$

3.1.3. Загрузите интерпретатор. Для этого вставьте кассету с программой и подайте команды: **C, 1200, СЛ**.

Если загорелся индикатор **ОМ** или **ОП**, повторите п.3.1.3.

3.1.4. Проверьте контрольную сумму кодов считанной программы, подав команду 1201. Контрольная сумма указана на кассете. При несовпадении контрольной суммы, а также при загорании индикатора **ОП** повторите п.п.3.1.3; 3.1.4.

3.1.5. Включите ПМ в автоматический режим, для чего нажмите до фиксации вторую слева вспомогательную кнопку (**TL2**) «Консул 260.1».

3.1.6. Запустите интерпретатор нажатием клавишей **C, S**. ПМ должна отпечатать:

**БЭЙСИК ДЗ-28, ВАРИАНТ 3
СНИМИТЕ КАСSETУ!**

3.1.7. Снимите кассету с интерпретатором и дайте подтверждение выполнения нажатием клавиши **ВК**.

3.1.8. Получив подтверждение, ПМ должна отпечатать:

НОМЕРА ВНЕШНИХ ПОДПРОГРАММ?

Оператор должен ввести с клавиатуры ПМ пустой или непустой список номеров внешних подпрограмм. Пустой список задаётся нажатием клавиши **ВК**.

Номерами внешних подпрограмм могут быть любые целые положительные числа от 1 до 7999. В списке номерá внешних подпрограмм отделяются друг от друга запятыми, заканчиваться список должен **ВК**.

3.1.9. Если список номеров внешних подпрограмм непустой, на печать выводится указание:

ПОСТАВЬТЕ КАСSETУ С ПОДПРОГРАММАМИ со списком номеров.

Оператор должен вставить кассету с внешними подпрограммами и подтвердить выполнение нажатием клавиши **ВК** (**⌘**).

Внешняя подпрограмма должна состоять из двух блоков в формате команды ДЗ-28 **SAVEX**. Первый блок длиной 8 байт должен содержать: номер подпрограммы в десятичной форме (2 байта), контрольную сумму, полученную командой **VERX**, записанную в шестнадцатеричной системе (2 байта), количество шагов в шестнадцатеричной системе (2 байта) и коды 0000 0512.

Второй блок содержит коды подпрограммы. Завершать его должны коды 0000 0512.

Указателем конца записей на ленте является подпрограмма с номером 7999. Если на ленте оказались не все подпрограммы из заданного списка, на печать вновь выводится указание:

ПОСТАВЬТЕ КАСSETУ С ПОДПРОГРАММАМИ

и перечисляются номера не найденных на предыдущих кассетах подпрограмм.

После того как все требуемые внешние подпрограммы прочитаны и загружены в память на печать выводится:

СНИМИТЕ КАСSETУ!

ГОТОВ

:

3.1.10. Если список номеров внешних подпрограмм пустой, на печать сразу выводится:

ГОТОВ

:

и интерпретатор готов к работе.

3.1.11. Загрузка интерпретатора может быть осуществлена также с перфоленты, отперфорированной программой **ПЕРФОРАЦИЯ В АБСОЛЮТНОМ ФОРМАТЕ** (см. И5М1.419.001 Д32), с помощью **АБСОЛЮТНОГО ЗАГРУЗЧИКА**. Порядок загрузки изложен в Руководстве оператора И5М1.419.001 Д33.

После загрузки интерпретатора в память выполняйте указания п.п. 3.1.4–3.1.10.

3.2. Режимы работы интерпретатора

3.2.1. Интерпретатор может работать как в режиме выполнения записанной в память Бэйсик-программы (программный режим), так и в режиме выполнения директив оператора, поступающих с клавиатуры пишущей машины («непосредственный» режим).

Признаком непосредственного режима интерпретатора и готовности к приёму директив оператора является двоеточие, выведенное на печать в левой крайней позиции. По завершению начального диалога интерпретатор находится в непосредственном режиме.

3.2.2. В непосредственном режиме оператор может:

а) построчно вводить или редактировать программу;

б) задать выполнение одного или нескольких операторов, записанных в одну строку;

в) запустить программу, т.е. перевести интерпретатор в программный режим.

Интерпретатор переходит из программного режима в непосредственный после выполнения операторов останова **END** и **STOP**, при выполнении последней строки программы, в большинстве случаев обнаружения ошибок, а также при прерывании, вызванном действиями оператора (см. **подраздел 3.4**).

Примечание. Оператор **INPUT** в непосредственном режиме интерпретатором не может быть правильно выполнен.

3.2.3. Интерпретатор отличает директиву ввода строки в текст программы от директивы исполнения операторов строки по наличию или отсутствию нóмера строки в её начале. При наличии нóмера строки её текст записывается в программу (см. [следующий подраздел](#)), при отсутствии – подлежит немедленному исполнению.

Концом ввода любой директивы оператора и сигналом к началу её исполнения интерпретатором является нажатие клавиши **БК** (или клавиши **ПС**). Пустые строки без номера интерпретатором игнорируются.

3.2.4. Выполнение операторов в непосредственном режиме используется в основном при загрузке Бэйсик-программ с перфоленты или магнитной ленты (оператор **LOAD**), печати листинга программы (оператор **LIST**), вспомогательных операций при отладке (вычисление значений выражений, печать значений переменных и т.п.) и вывода текста отлаженной программы на носители (оператор **SAVE**).

Однако в некоторых случаях в непосредственном режиме могут быть выполнены довольно значительные по объёму вычисления. Например, одной строкой

```
FOR X=0 TO #PI/2 STEP .01: PRINT !0.6! X;SIN(X): NEXT X
```

задаётся процедура вычисления и печати таблицы значений функции **SIN(X)**.

3.3. Набор, редактирование и запуск программы

3.3.1. Программа пользователя набирается на клавиатуре ПМ «Консул 260.1» построчно. Каждая строка должна начинаться номером строки и заканчиваться символом **БК** (**ПС**). Длина строки не должна превышать 100 символов.

3.3.2. Если во время ввода строки допущена ошибка и она замечена до нажатия клавиши **БК**, её можно исправить, пользуясь клавишей **▣**. Эта клавиша используется в Бэйсике в качестве клавиши забоя. Каждое нажатие клавиши **▣** уничтожает один символ в уже набранном тексте строки, что позволяет затем ввести вместо неправильных символов правильные.

Нажав клавишу **U** нижнего регистра при нажатой первой вспомогательной кнопке ПМ слева (**TL1**), можно стереть всю набираемую строку (печатается сообщение **→U**) и начать ввод с начала строки, предварительно отжав кнопку.

3.3.3. Изменить строку, уже записанную в программу, можно только повторным набором.

Для того чтобы исключить строку из программы, надо набрать её номер и нажать клавишу **БК** (**ПС**). Исключить одну или несколько строк из программы можно также с помощью оператора **CLEAR** (см. [п. 2.14.2](#)).

3.3.4. Вставить строку можно только между строками, номерá которых не являются последовательными числами. Вставляемой строке надо присвоить номер, промежуточный между номерами раздвигаемых строк, и набрать её на клавиатуре ПМ «Консул 260.1».

3.3.5. Запуск программы пользователя осуществляется подачей в непосредственном режиме, в строке без номера, одного из операторов: **RUN**, **GOTO** или **GOSUB**.

Для операторов **GOTO** и **GOSUB** вслед за именем оператора необходимо указать номер строки, с которой запускается программа (**GOTO**) или подпрограмма (**GOSUB**).

При подаче команды **RUN** выполнение программы начинается со строки с минимальным номером. При запуске программы пользователя командой **RUN** обнуляются указатели циклов и подпрограмм, указатель данных для **READ** и стираются имеющиеся в памяти данные (области **COM** и **DIM** памяти).

Следует иметь в виду, что при запуске программы операторами **GOTO** или **GOSUB** сохраняются все определённые на этот момент переменные и состояния указателей цикла и подпрограмм, что может привести к сообщениям об ошибках, связанных с повторным определением массивов, превышением допустимого уровня вложенности и т.п.

3.4. Остановы программы

3.4.1. Все остановки Бэйсик-программы осуществляются с печатью сообщения **ОСТАНОВ В СТРОКЕ** с указанием номера строки и переходом в непосредственный режим, с печатью двоеточия в начале следующей строки.

Останов может быть вызван операторами **STOP** и **END**, выполнением последнего предложения программы, ошибкой в программе или прерыванием, вызванным оператором.

Останову по ошибке предшествует сообщение об ошибке (см. подраздел 3.5), а останову по прерыванию – сообщение **↵P**.

3.4.2. Прервать выполнение программы можно двумя способами:

а) нажав клавишу **P** ПМ в латинском регистре при нажатой левой вспомогательной кнопке (**TL1**);

б) нажатием до фиксации средней вспомогательной кнопки (**TL3**).

После печати сообщения об останове кнопки следует отпустить.

3.5. Сообщения об ошибках

3.5.1. При возникновении ошибки при вводе или исполнении программы пользователя интерпретирующая программа выдаёт на печать сообщение об ошибке в форме:

ОШИБКА <номер ошибки> **В СТРОКЕ** <номер строки>

Для ошибок непосредственного режима, а также для ошибок в строке, вводимой по оператору **INPUT**, и при вводе с перфоленты будет указан нулевой номер строки.

Перечень ошибок приведён в табл. 2. При вводе программы пользователя возможно возникновение ошибок 0, 2, 4, 5, 6 (см. табл. 2). Остальные ошибочные ситуации анализируются в процессе выполнения операторов и программы.

3.5.2. Ошибки с номерами 0–99 вызывают останов программы и переход в непосредственный режим.

При ошибках с номерами 121–128 после печати сообщения об ошибке выполнение программы продолжается.

Таблица 2

Номер ошибки	Содержание ошибки
0	Переполнение памяти, отведённой пользователю.
1	Недопустимый оператор.
2	Переполнение строки ввода.
3	Недопустимый ограничитель в строке.
4	Недопустимый номер строки.
5	Несоответствие кавычек в предложении.
6	Отсутствие открывающей скобки перед аргументом функции.
7	Недопустимый оператор LET .
10	Неправильная запись индексов.
11	Неправильная размерность индекса; массив не определён.
12	Несоответствие скобок в выражении.
13	Недопустимый элемент выражения.
14	Функция пользователя не определена.
15	Неправильное имя переменной.
20	Неправильная операция отношения.
21	Недопустимый оператор IF .
22	Неправильный DIM, COM .
23	Недостаточно места для массива.
24	Неправильный DEF .
25	Нет данных для READ .
26	Недопустимый оператор DATA .
27	Неправильный формат команд в CMD .
30	Неправильный формат FOR – NEXT .
31	Нет NEXT .
32	Не было FOR .
33	Переполнение стека FOR – NEXT .
34	Нулевой шаг FOR .
35	Неправильный формат PRINT .
36	Неправильно задан формат печати.
37	Недопустимое выражение в TAB .
38	Отсутствие открывающей записи в буфере МЛ.
40	$HC_2 < HC_1$.
41	Превышение уровня подпрограмм.
42	RETURN без GOSUB .
43	Нет строки для перехода по GOSUB или GOTO .
44	Нет внешней подпрограммы с указанным номером.
50	Неправильное предложение с операторами обслуживания МЛ и перфоленты.
52	Сбой структуры файла.
53	Отсутствие в ЗУ массива при приёме с МЛ и перфоленты.
54	Не считан очередной блок данных с МЛ.
55	Считанный с МЛ блок не помещается в ОЗУ.
59	При загрузке (или записи) программы с МЛ указан только один номер строки.
121	Недопустимые знаки при вводе по INPUT .
122	Недостаточно данных для INPUT .
123	Несуществующая переменная.
124	Слишком много данных для INPUT .
128	Некорректная операция (OP) в процессе вычислений.

3.6. Рекомендации по отладке программ

3.6.1. Простые Бэйсик-программы могут при первом же запуске выдавать правильные результаты. Более сложные программы обычно требуют поиска и устранения ошибок.

3.6.2. Ошибки, связанные с неправильным вводом текста программы, обычно легко обнаруживаются путём сличения листинга, полученного по оператору **LIST**, с исходным текстом. Редактирование программы сводится к повторному вводу строк, содержащих ошибки.

3.6.3. Для обнаружения более сложных логических ошибок приходится вставлять в программу вспомогательные операторы, чаще всего операторы печати **PRINT**, для вывода промежуточных результатов счёта, выходных значений переменных подпрограмм, трассировки хода процесса и т.п. После отладки эти операторы, естественно, удаляются.

3.6.4. Для облегчения понимания сообщений об ошибках, несмотря на несколько больший требуемый для программы объём памяти и большее время счёта, рекомендуется записывать в строке только одно предложение. После отладки в целях улучшения качества программы строки без обращения можно объединить.

3.7. Остановы интерпретатора

3.7.1. В процессе выполнения программы пользователя через каждые 255 строк производится контроль сохранности в ЗУ интерпретирующей программы. При изменении контрольной суммы интерпретирующей программы или контрольной суммы таблицы подпрограмм программа-интерпретатор останавливается, что говорит или о неисправности ДЗ-28, или о неосмотрительных действиях пользователя, например, при применении операторов **CMD**, **CALL**.

3.7.2. При чтении с МЛ блока недопустимой длины (более 256 байт) интерпретатор может остановиться с индикацией $X = 55$. Как правило, для продолжения работы требуется повторная загрузка и запуск интерпретатора.

3.7.3. При необходимости работа интерпретатора может быть остановлена клавишей **Ш** и запущена снова клавишами **▷**, **М** с клавиатуры ДЗ-28.

4. ДОПОЛНИТЕЛЬНЫЕ СВЕДЕНИЯ

4.1. Распределение памяти

4.1.1. После загрузки программа расположена в ОЗУ с адреса 0.00.00.00 по адрес 2.02.02.13. Распределение ОЗУ непосредственно после загрузки показано на **рис. 1**.

4.1.2. По запуску программы управление передаётся программе начального диалога, которая запрашивает у оператора сведения о необходимости использования внешних подпрограмм, загружает внешние подпрограммы, устанавливает указатели,

переписывает таблицу адресов подпрограмм на адреса с X.13.00.00 по X.13.15.15. Здесь и далее X = 3 для ОЗУ 16 Кбайт, X = 7 для ОЗУ 32 Кбайт.

0.00.00.00	ИНТЕРПРЕТАТОР
2.00.00.00	Таблица адресов внутренних подпрограмм
2.01.00.00	Программа начального диалога
2.02.02.14	Свободная зона ОЗУ

Рис. 1. Распределение памяти непосредственно после загрузки.

0.00.00.00	ИНТЕРПРЕТАТОР	
2.00.00.00	Внешние подпрограммы	
(T7)=(X.13.08.12)	Область пользователя	↓

	Стек	↑
(R13)=(X.13.11.00)	Таблица адресов внешних подпрограмм	
(X.13.08.14)=X.11.00.00	Служебная память интерпретатора	
X.13.00.00	Таблица адресов внутренних подпрограмм	
X.14.00.00	Служебная зона интерпретатора	
X.15.00.00	Служебная зона ДЗ-28	

Рис. 2. Распределение ОЗУ после начального диалога.

В процессе начального диалога внешние подпрограммы загружаются последовательно друг за другом в свободную часть ОЗУ, начиная с адреса 2.03.00.00.

После завершения загрузки внешних подпрограмм таблица адресов внутренних подпрограмм и программа начального диалога стираются, а на их место переписываются внешние подпрограммы, начиная с адреса 02.00.00.00.

Начальные адреса внешних подпрограмм записываются в таблицу, нижней границей которой является X.11.00.00 – адрес, указанный в ячейке X.13.08.14 таблицы внутренних подпрограмм.

Каждой внешней подпрограмме в таблице соответствует ячейка из четырёх байтов. Два старших байта занимает номер внешней подпрограммы в десятичной форме, два следующих – содержат начальный адрес внешней подпрограммы в шестнадцатеричной форме.

Указатель начала программы пользователя, расположенный в ячейке X.13.08.12, устанавливается на адрес первого свободного шага после внешних подпрограмм, а начальное состояние указателя стека (R13), соответствующее верхней границе таблицы адресов внешних подпрограмм, – в ячейке X.13.11.10.

Если внешних подпрограмм нет, указатель начала программы пользователя устанавливается равным 02.00.00.00, а начальное состояние указателя стека – X.11.00.00.

Распределение памяти после начального диалога показано на [рис. 2](#).

4.1.3. В области пользователя хранится текст исходной программы, данные (переменные и массивы) и указатели функций пользователя. В общем случае структура области пользователя при работе интерпретатора имеет вид, приведённый на [рис. 3](#). Заполнение области пользователя и стека производится во встречных направлениях.

(T7)	Текст Бэйсик-программы	
(T2)	Область COM (переменные и массивы, объявленные оператором COM , указатели функций пользователя FN)	Расширение области производится вставкой в начало
(T3)	Область DIM (массивы, объявленные оператором DIM , простые переменные, определённые любыми операторами, кроме оператора COM)	Расширение области производится вставкой в конец
(R3)	Резервная зона	
(R13)		

Рис.3. Структура области пользователя.

4.1.4. Служебная память интерпретатора состоит из буфера МЛ (X.11.00.00–X.11.15.15) и буфера ввода строки (X.12.00.00–X.12.15.15).

Использование таблицы адресов внутренних подпрограмм приведено на [рис. 4](#). Служебная зона интерпретатора состоит из стека подпрограмм (адреса X.14.04.00–X.14.03.15), стека циклов (X.14.04.00–X.14.14.15) и двух десятичных ячеек генератора случайных чисел (адреса X.14.15.00–X.14.15.15).

4.2. Общие принципы работы интерпретатора. Регистры

4.2.1. Упрощённый алгоритм работы интерпретатора приведён на [рис. 5](#).

Выход на подпрограмму реализации оператора производится по его коду (см. [подраздел 4.3](#) и [рис. 4](#)).

4.2.2. Сведения об использовании интерпретатором регистров ДЗ-28 приведены в [табл. 3](#).

4.3. Представление исходной программы

4.3.1. Слова исходной программы, введённые при вводе программы посимвольно, перекодируются интерпретатором в соответствии с [таблицей 4](#).

Символы строк, не входящие в слова Бэйсика, не изменяются.

4.3.2. Введённый код возврата каретки (0013) заменяется интерпретатором на код перевода строки (0010).

	00	02	04	06	08	10	12	14
X.13.00.00	CLEAR C	ВЫВОД ВК	ВЫБОРКА	ПОИСК ПЕР		ВВОД СТР	УСТ АДР	ЧТЕН ИЗ СТР
01	ЦЕЛОЕ	SAVE #1	ИМЯ 2	ЧТЕН ЧИСЛ		ЧТЕН НС	ЗАП ЗНАЧ	ЗАП ПЕР
02	ВЫЗОВ СТЕК	ИЗОБР	ВЫЗОВ ЗНАЧ	КОД СЛОВА	ЗАП СОМ	ПОИСК ПС	ПРОВ ИНД	ЗАП СТЕК
03	ПЕЧ ОШИБ	ПОИСК КОП	"В СТРОКЕ"	IF END		ЗАП (R10)	ВЫЧИСЛ АДР	ИСКЛ СИМВ
04		GOTO 1	ПОДГ "0"	АНАЛИЗ СТРОКИ		ОВЗ	ВЫВ ⊐	ВЫЗОВ ПЕР
05	ЗАП В ПРОГ	ПОДГ ПЕЧ	ОПР ГРАН	ИЗМ ГРАН		ПОИСК ИМЁН	ВЫВОД ВШ	ВСТАВИТЬ
06	DATA SAVE	DATA LOAD	ВЫВ *	ВЫВОД СИМВ	АДР ЭЛЕМ	ЧТЕН АРГ	ИМЯ ПЕР	ФОРМ ПЕР
07	АНАЛИЗ СИМВ	ОС	ПОИСК СТР	СТИР СТР	ВЫВ ПУС	CLEAR D	ВЫЧИСЛ	ВВ/ВЫВ←0
08	ФУНКЦ	ОБР СТР	ОПР ПОЛ КАР	ВЫВ ВР	RND	УПАК СТР	НПР	НУС
09	ЗАП ЗАП	ЧТЕН ЗАП	ПРИЁМ ИМЁН	ОЧИСТКА	ПОИСК ИМФ	АНАЛИЗ НС	ПОДГ МАС	DATA SAVE #1
10	DATA LOAD #1	LOAD #1	МАС ПРИЁМА	ГРАН МАС	ПЕРЕЗАПИСЬ	ПРИЁМ БАЙТА	АНАЛИЗ БАЙТА	CLEAR P
11		КС ПР1	КС ПР2	КС ТАБЛ	7.15.15.15	УС РН	АНАЛИЗ П4	ОБЪЁМ ОЗУ
12	LET	READ	GOTO	GOSUB	RETURN	IF	FOR	NEXT
13	INPUT	DATA	DIM	COM	PRINT	LOAD	SAVE	
14	CMD		SKIP			REWIND	DEF	END
15	STOP	LIST	REM	RUN	RESTORE	CLEAR	CALL	ON

Рис.4. Таблица имён внутренних подпрограмм.

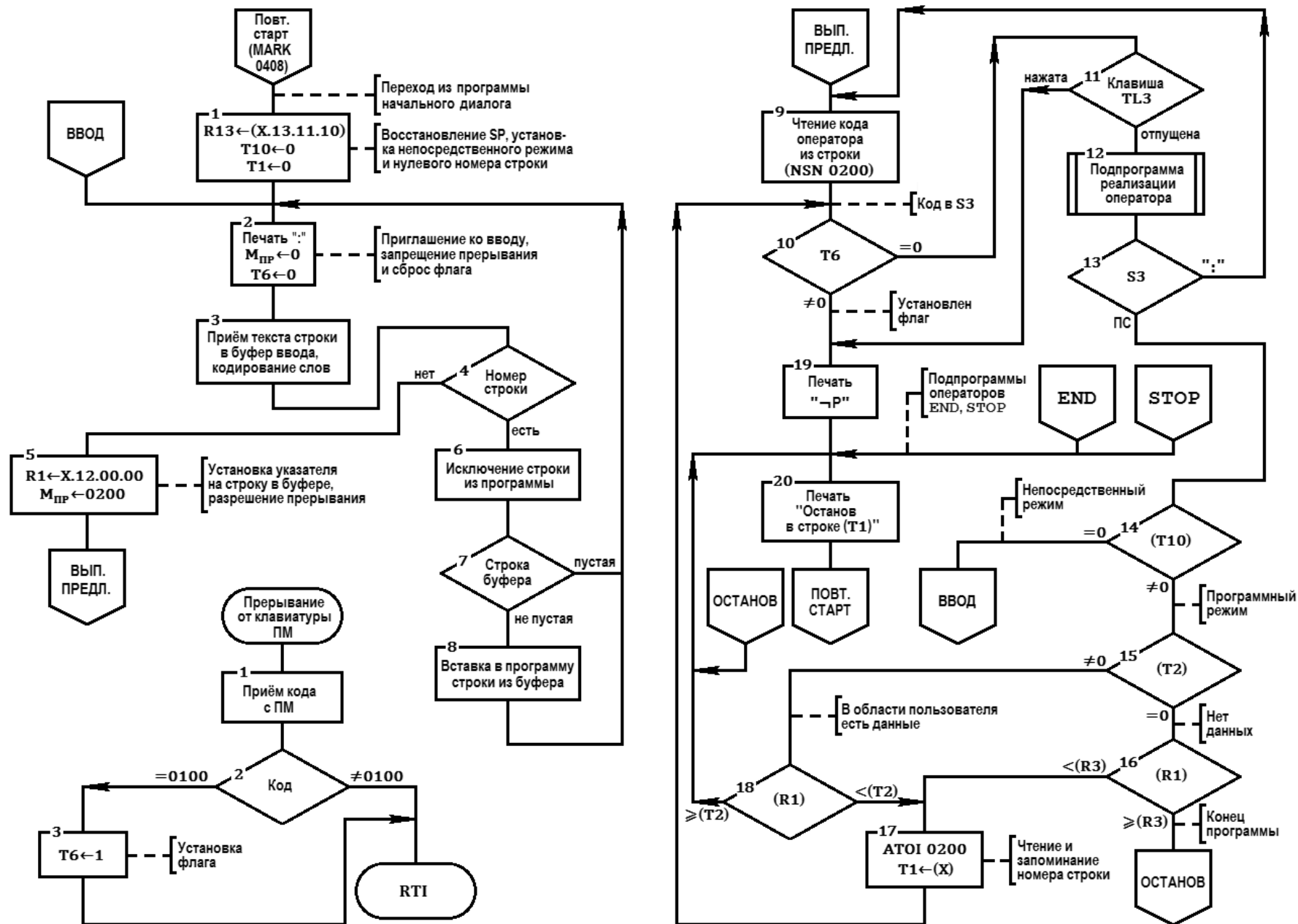


Рис.5. Алгоритм работы интерпретатора.

Таблица 3

Регистр	Назначение содержимого
R0	Константа нуль
R1	Текущий адрес в тексте программы
R2	Константа два
R3	Указатель границы области пользователя
R13	Указатель стека
S3	Очередной символ текста программы
T1	Номер текущей строки
T2	Начало областей данных
T3	Конец области COM
T4	Указатель блока DATA в тексте программы
T5	Счётчик позиций печати
T6	Флаг прерывания
T7	Начало программы пользователя
T8	Указатель стека GOSUB
T9	Указатель стека циклов FOR-NEXT
T10	Режим интерпретатора
T11	Текущий номер записи на МЛ
T14	Формат печати PRINT
T15	Счётчик строк (для контроля сохранности интерпретатора)

Таблица 4

Слово	Код	Слово	Код
CALL	1512	HSN(0808
CLEAR	1510	HTN(0812
CMD	1400	IF	1210
COM	1306	INPUT	1300
COS(0902	INT(1004
DATA	1302	LET	1200
DEF	1412	LGT(1006
DEG(0800	LIST	1502
DIM	1304	LOAD	1310
END	1414	LOG(1008
EXP(1002	NEXT	1214
EXT(1010	ON	1514
FOR	1212	OPEN	1106
FN	1104	PRINT	1308
GOSUB	1206	READ	1202
GOTO	1204	REM	1504
HCS(0910	RESTORE	1508
RETURN	1208	STEP	1102
REWIND	1410	STOP	1500
RND(1012	SQR(0908
RUN	1506	TAB	1110
SKIP	1404	#PI	1014
SAVE	1312	TAN(0804
SGN(0808	THEN	1108
SIN(0802	TO	1100

4.4. Данные и служебные записи

4.4.1. Простые переменные и массивы хранятся в области данных (безразлично, в DIM или COM) в виде записей однотипного формата.

Каждая запись состоит из имени (два байта), определителя (два байта) и необходимого числа восьмибайтовых десятичных ячеек для хранения значений элементов. Имя записи состоит из кода буквы и кода цифры имени.

Отсутствующая в имени переменной или массива цифра заменяется кодом 0000.

Определитель записи состоит из двух байтов, содержащих в шестнадцатеричном коде максимальные значения индексов массивов, причём старший байт соответствует первому индексу. Общий вид формата записи массива приведён на рис. 6.

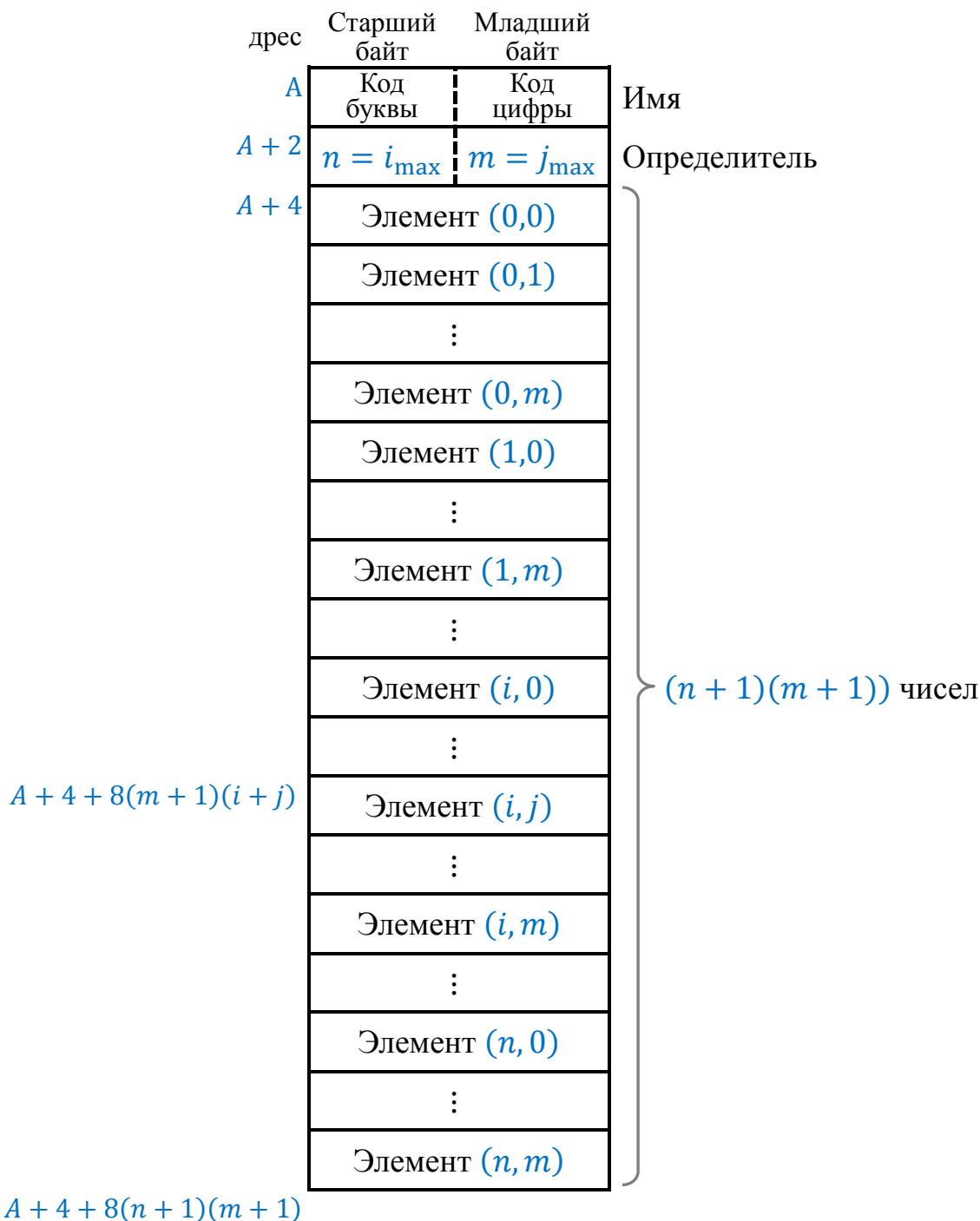


Рис.6. Формат записи массива.

На рис. 7-9 в качестве примера приведён формат записи в области данных одномерного массива **C1(3)**, двумерного массива **B(1,3)** и простой переменной **A9**.

0403	0301	Имя C1 Определитель 3,0
0003	0000	
C1(0) = C1		
C1(1)		
C1(2)		
C1(3)		

Рис. 7. Запись массива **C1(3)**.

4.4.2. Кроме записей массивов, объявленных оператором **COM**, в одноимённую область данных заносятся записи-указатели функций пользователя.

0401	0309	Имя A9 Определитель 0,0
0000	0000	
A9		

Рис. 8. Запись простой переменной **A9**.

Каждая запись состоит из шести байтов – имени функции пользователя, формального параметра и адреса определения функции в тексте программы. За последний принят абсолютный адрес первого символа, следующего за знаком равенства в операторе **DEF**.

Структура записи-указателя функции пользователя приведена на рис. 10.

0402	0000	Имя B Определитель 1,3
0001	0003	
B(0,0) = B(0) = B		
B(0,1) = B(1)		
B(0,2) = B(2)		
B(0,3) = B(3)		
B(1,0)		
B(1,1)		
B(1,2)		
B(1,3)		

Рис. 9. Запись двумерного массива **B(1,3)**.

4.4.3. Структура служебной записи в стеке **FOR-NEXT** приведена на рис. 11. Запись занимает 22 байта. Служебная запись в стеке **GOSUB** содержит адрес разделителя предложения **GOSUB** (два байта).

4.5. Внутренние подпрограммы

4.5.1. В данном подразделе описаны внутренние подпрограммы интерпретатора, которые могут быть использованы пользователем в своих внешних подпрограммах.

Обращение к этим подпрограммам осуществляется с помощью команд **JSTT** (1013 **B₂A₂**), где **B₂A₂** – код подпрограммы.

Все внутренние подпрограммы рассчитаны на **(BD) = 0** и **(BP) = 0**.

4.5.2. Подпрограмма **ВЫЧИСЛ** (**JSTT** 0712) производит вычисление значения арифметического выражения, записанного в указанном месте ОЗУ в кодах ПМ «Консул 260.1» без дополнения до чётности.

Перед обращением к подпрограмме **ВЫЧИСЛ** в **R1** должен быть указан адрес первого символа вычисляемого арифметического выражения.

Все переменные и функции пользователя, используемые в арифметическом выражении, долж-

Код (1104)	Код буквы	Имя функции
Формальный параметр		
Адрес определения		

Рис. 10. Формат записи-указателя функции пользователя.

(T9)	имя управляющей переменной	2 байта
	адрес определителя управляющей переменной	2 байта
	адрес разделителя предложения FOR	2 байта
	граничное значение управляющей переменной	8 байтов
	шаг цикла	8 байтов

(T9)+

Рис. 11. Структура служебной записи в стеке **FOR-NEXT**.

ны быть определены до обращения к подпрограмме вычисления. Результат вычислений заносится в регистр **X**.

По окончании вычисления в **S3** заносится первый символ, не входящий в арифметическое выражение, а (**R1**) указывает на следующий за ним символ.

Подпрограмма использует все регистры ДЗ-28.

4.5.3. Подпрограмма **ВЫВОД СИМВ** (JSTT 0606) выводит посимвольно информацию из ОЗУ, начиная с адреса, указанного в **R10**, и до адреса, на котором записан код 0000. В зависимости от состояния регистра **T10**, вывод информации производится или на печать или на перфоленту. Если младший байт (**T10**) равен нулю, то вывод информации ведётся на ПМ, в противном случае вывод производится на перфоленту.

4.5.4. Подпрограмма **ВВОД СТР** (JSTT 0010) принимает посимвольно информацию или с клавиатуры ПМ, или с перфоленты, в зависимости от состояния младшего байта регистра **T10**.

Если младший байт **T10** равен нулю, ввод производится с ПМ «Консул 260.1», в противном случае – с перфоленты. Приём информации производится в буфер ввода, начинающийся с адреса X.12.00.00.

Приём заканчивается приёмом кодов 0013 (**БК**) или 0010 (**ПС**). Старший бит принятого кода (контроль на чётность) при приёме обнуляется.

Подпрограмма использует регистры **R12**, **R10**, **R9**.

4.5.5. Подпрограмма **АДР ЭЛЕМ** (JSTT 0608) ищет в области данных переменную или массив с именем, указанным в **R9**. Если переменной или массива с указанным именем нет, в **R8** заносится нуль.

R1 перед обращением к подпрограмме должен указывать на символ, следующий в программе за именем переменной. Если за именем переменной следует индексное выражение в скобках (задан элемент массива), производятся чтение индексных выражений и вычисление адреса указанного элемента. Уменьшенное на 2 значение адреса элемента массива заносится в **R8**, в **S3** заносится первый символ после закрывающей скобки, **R1** указывает на следующий адрес.

Если задана простая переменная, в **R8** заносится адрес её определителя, в **S3** заносится первый символ после имени переменной, **R1** указывает на следующий адрес.

При поиске простой переменной используются регистры **R4**, **R8**, **R9**, **R10**, **R11**, **R12**, при поиске элементов массива используются все регистры.

4.5.6. Подпрограмма **ЗАП ЗНАЧ** (JSTT 0112) записывает в восьмибайтовую ячейку по адресу, равному $(R8) + 2$, значение переменной из регистра **X**. После выполнения подпрограммы в **R8** заносится $(R8) + 8$, **R8** указывает на адрес последней 2-байтовой ячейки, занятой записанным числом.

Подпрограмма использует регистры **R4**, **R5**, **R6**, **R7**, **R8**.

4.5.7. Подпрограмма **ВЫЗОВ ПЕР** (JSTT 0414) ищет в ОЗУ переменную или массив с именем, указанным в **R9**, и вызывает в регистр **X** значение указанной переменной. Если переменной с указанным именем в области данных нет, выводится сообщение об ошибке 123.

R1 перед обращением к подпрограмме **JSTT 0414** должен указывать на первый символ после имени переменной.

После выполнения подпрограммы в **S3** заносится первый символ после имени переменной или после закрывающей скобки (при задании элемента массива), **R1** указывает на следующий адрес.

Подпрограмма использует все регистры.

4.5.8. Подпрограмма **ЗАП СТЕК (JSTT 0214)** записывает в стек, указателем которого является **R13**, число из регистра **X** в восьмибайтовом десятичном формате. Указатель стека при этом пересчитывается, в **R13** заносится значение $(R13) - 8$.

Подпрограмма использует регистры **R4, R5, R6, R7, R10**.

4.5.9. Подпрограмма **ВЫЗОВ СТЕК (JSTT 0200)** вызывает из стека, указателем которого является **R13**, в регистр **X** число, которое было записано в стеке в восьмибайтном десятичном формате. Указатель стека при этом пересчитывается, в **R13** заносится значение $(R13) + 8$.

Подпрограмма использует регистры **R4, R5, R6, R7, R10**.

4.5.10. Подпрограмма **ФОРМ ПЕР (JSTT 0614)** формирует в ОЗУ новую простую переменную с именем, указанным в **R10**. Переменная формируется по адресу старой границы области данных $(R3)$, в **R3** при этом заносится новое значение $(R3) + 12$.

Значение переменной и её определитель принимаются равными нулю. Подпрограмма использует регистры **R8** и **R10**.

4.5.11. Подпрограмма **ОБЪЁМ ОЗУ (JSTT 1114)** вызывает в **S9** указатель объёма ОЗУ, который равен 0300 для ОЗУ 16 Кбайт и 0700 для ОЗУ 32 Кбайт.

Подпрограмма использует регистр **R12**.

4.5.12. Подпрограмма **ИМЯ ПЕР (JSTT 0612)** читает из программы имя переменной, начиная с адреса, указанного в **R1**, и заносит это имя в **R9**. Если **R1** указывает не на латинскую букву, на печать выводится сообщение об ошибке 15. После выполнения подпрограммы **R1** указывает на первый символ, следующий за именем переменной.

Подпрограмма использует регистры **R8** и **R9**.

4.5.13. Подпрограмма **ПОИСК ПЕР (JSTT 0006)** читает из текста программы имя переменной, начиная с адреса, указанного в **R1**, ищет в области данных переменную или массив с указанным именем, и заносит в **R8** адрес определителя этой переменной. Если переменной или массива с указанным именем в области данных нет, **R8** обнуляется. Если задана индексированная переменная, то в **R8** заносится уменьшенный на 2 адрес указанного элемента массива.

После выполнения подпрограммы в **R10** заносится имя переменной, в **S3** – символ, следующий за именем простой переменной или её закрывающей скобкой для элементов массива, а **R1** указывает на следующий адрес.

Подпрограмма использует все регистры.

4.6. Простейшие возможности изменения интерпретатора

4.6.1. В интерпретаторе предусмотрена довольно значительная программная выдержка времени на исполнение команды возврата каретки пишущей машины.

При правильной регулировке контроллера и механизма ПМ в целях ускорения вывода эту выдержку времени можно исключить или уменьшить изменением интерпретатора на шаге 2601.

4.6.2. Изменением шага 4133 можно заменить символ, используемый в качестве заобоя.

4.6.3. При желании можно исключить совсем или изменить периодичность контроля сохранности интерпретатора.

Для того чтобы исключить контроль сохранности интерпретатора, необходимо на шаге 375 записать код 0108. Для изменения периодичности контроля необходимо изменить информацию на шагах 431 и 433 (количество строк в шестнадцатеричной системе).

4.6.4. При необходимости на шагах 00016–00047 могут быть размещены подпрограммы обработки внешних прерываний ПР8–ПР1. Для обмена данными между прерывающими программами и внешними подпрограммами могут быть использованы регистры Т12 и Т13.

4.6.5. Естественно, все изменения интерпретатора должны быть внесены до его запуска.

5. ПРИМЕРЫ ПРОГРАММ

5.1. Корни квадратного уравнения

5.1.1. Программа вычисления корней квадратного уравнения $Ax^2 + Bx + C = 0$, заимствованная из [1] с весьма незначительными изменениями, приведена на рис. 12. Программа проста и не требует дополнительных пояснений.

Пример протокола работы программы приведён на рис. 13.

5.1.2. По рис. 12 можно отметить следующие недостатки программы:

- а) слишком «широкая» печать;
- б) для повторения программы с новыми коэффициентами требуется замена блока данных в повторный запуск.

Незначительными изменениями рассмотренная выше программа превращается в циклическую (рис. 14), свободную от указанных недостатков. Выход из цикла осуществляется по вводу нулевых коэффициентов.

Ввод каждого коэффициента завершается символом \.

5.2. Биоритмы

5.2.1. На рис. 15 приведён текст программы вывода графиков т.н. биологических ритмов человека:

Φ – «физического» (период 23 дня);

```

LIST
1 PRINT !E!                               ← установка формата
10 DATA 3.5,7.64,-0.25E-4                 ← коэффициенты
20 READ A,B,C
30 IF A<>0 THEN 120
40 IF B<>0 THEN 100
50 IF C<>0 THEN 80
60 PRINT 'РЕШЕНИЙ БЕСКОНЕЧНО МНОГО'
70 GOTO 7999
80 PRINT 'РЕШЕНИЙ НЕТ'
90 GOTO 7999
100 PRINT 'КОРЕНЬ ОДИН X=';-C/B
110 GOTO 7999
120 LET E=2*A
130 LET D=B-2-2*E*C
140 IF D<>0 THEN 170
150 PRINT 'КОРНИ КРАТНЫЕ X1=X2=';-B/E
160 GOTO 7999
170 LET D1=SQR(ABS(D))
180 IF D<0 THEN 220
190 PRINT 'КОРНИ ДЕЙСТВИТЕЛЬНЫЕ X1=';(-B+D1)/E,
200 PRINT 'X2=';(-B-D1)/E
210 GOTO 7999
220 PRINT 'КОРНИ КОМПЛЕКСНЫЕ СОПРЯЖЕННЫЕ'
230 PRINT 'X1=';-B/E;' +I*';ABS(D1/E),
240 PRINT 'X2='; -B/E;' -I*';ABS(D1/E)
7999 END
:

```

Рис. 12. Программа вычисления корней квадратного уравнения.

```

RUN
КОРНИ ДЕЙСТВИТЕЛЬНЫЕ X1= .318877000000E-05   X2= -.224000318875E 01

ОСТАНОВ В СТРОКЕ   7999
:10DATA 1,2,1       ← замена блока данных
:RUN
КОРНИ КРАТНЫЕ X1=X2= -.100000000000E 01

ОСТАНОВ В СТРОКЕ   7999
:DATA 0,1,8
:10 DATW#A 0,1,8    ← забой ошибочного символа
:RUN
КОРЕНЬ ОДИН X= -.800000000000E 01

ОСТАНОВ В СТРОКЕ   7999
:10 DATA 1,0,1
:RUN
КОРНИ КОМПЛЕКСНЫЕ СОПРЯЖЕННЫЕ
X1= -.000000000000   +I*.100000000000E 01   X2= -.000000000000 -I*
.100000000000E 01   ← не хватило места для печати

ОСТАНОВ В СТРОКЕ   7999
:

```

Рис. 13. Пример протокола работы программы 5.1.

```

LIST
1 PRINT !F1.9!
10 INPUT ' 'A
15 INPUT '*X-2+'B
20 INPUT '*X+'C
25 PRINT '=0'
30 IF A<>0 THEN 120
40 IF B<>0 THEN 100
50 IF C<>0 THEN 80
60 STOP
70 GOTO 7999
80 PRINT 'РЕШЕНИЙ НЕТ'
90 GOTO 7999
100 PRINT 'КОРЕНЬ ОДИН X=';-C/B
110 GOTO 7999
120 LET E=2*A
130 LET D=B-2-2*E*C
140 IF D<>0 THEN 170
150 PRINT 'КОРНИ КРАТНЫЕ X1=X2=';-B/E
160 GOTO 7999
170 LET D1=SQR(ABS(D))
180 IF D<0 THEN 220
190 PRINT 'КОРНИ ДЕЙСТВИТЕЛЬНЫЕ X1=';(-B+D1)/E,
195 PRINT: PRINT TAB 21
200 PRINT 'X2=';(-B-D1)/E
210 GOTO 7999
220 PRINT 'КОРНИ КОМПЛЕКСНЫЕ СОПРЯЖЕННЫЕ'
230 PRINT 'X1=';-B/E;' +I*';ABS(D1/E),
235 PRINT
240 PRINT 'X2=';-B/E;' -I*';ABS(D1/E)
7999 GOTO 10
:

RUN
3.5*X-2+7.84*X+-.25E-4=0
КОРНИ ДЕЙСТВИТЕЛЬНЫЕ X1= 3.188770000E-06
X2= -2.240003189

1*X-2+2*X+1=0
КОРНИ КРАТНЫЕ X1=X2= -1.000000000
0*X-2+1*X+8=0
КОРЕНЬ ОДИН X= -8.000000000
1*X-2+0*X+1=0
КОРНИ КОМПЛЕКСНЫЕ СОПРЯЖЕННЫЕ
X1= -.000000000 +I* 1.000000000
X2= -.000000000 -I* 1.000000000
0*X-2+0*X+0=0

ОСТАНОВ В СТРОКЕ 60
:

```

Рис. 14. Циклическая программа вычисления корней квадратного уравнения.

```

LIST
10 GOTO 190: DATA 31,28,31,30,31,30,31,31,30,31,30,31
15 REM ПОДПРОГРАММА СОРТИРОВКИ МЕТОДОМ ШЕЛЛJA
20 FOR I=0 TO B-L
30 LET J=I
40 IF P(J)<=P(J+L) GOTO 80
50 LET A=P(J): LET P(J)=P(J+L): LET P(J+L)=A
60 LET A=Q(J): LET Q(J)=Q(J+L): LET Q(J+L)=A
70 IF J>=L THEN LET J=J-L: GOTO 40
80 NEXT I: IF L>1 THEN LET L=(L-1)/2: GOTO 20
90 RETURN
95 REM ПОДПРОГРАММА ОПРЕДЕЛЕНИЯ ВИСОКОСНОСТИ
100 LET A1=INT(G/100): LET A2=G-A1*100: LET A=A2/4
110 IF A2=0 THEN LET A=A1/4
120 IF A=INT(A) THEN LET M(2)=29: LET N1=1: RETURN
130 LET M(2)=28: LET N1=2: RETURN
135 REM ПОДПРОГРАММА СЧЕТА КОДА ДНЯ
140 GOSUB 100
150 LET C=INT(365.25*G)+INT(30.56*M)+D
160 IF M<3 THEN LET C=C+N1
170 RETURN
195 REM ВХОД В ПРОГРАММУ
200 DIM P(9),Q(9),M(12)
205 FOR I=1 TO 12: READ M(I): NEXT I
210 DEF FNP(E)=20*(SIN(2*#PI*C/E))
230 INPUT 'ДАТА.МЕСЯЦ, ГОД РОЖДЕНИЯ           'R,G
240 INPUT 'ДАТА.МЕСЯЦ, ГОД НАЧАЛА ГРАФИКА 'N,G1
250 INPUT 'ДАТА.МЕСЯЦ, ГОД КОНЦА ГРАФИКА  'K,G2
270 LET D=INT(R): LET M=(R-D)*100: GOSUB 140
280 LET C1=C: LET G=G2: LET D=INT(K): LET M=(K-D)*100: GOSUB 140
290 LET C2=C-C1: LET G=G1: LET D=INT(N): LET M=(N-D)*100: GOSUB 140
315 REM ЧИСЛО РИТМОВ И НАЧ. ШАГ СОРТИРОВКИ
320 PRINT: LET B=3: LET L=1
325 REM ТЕЛО ПРОГРАММЫ
330 FOR C=C-C1 TO C2
340 PRINT !2.2! D+M/100 !1.0!
350 LET P=0: LET P(1)=FNP(28)
360 LET P(2)=FNP(23): LET P(3)=FNP(33)
380 LET D=D+1
390 IF D<=M(M) GOTO 450
400 LET D=1: LET M=M+1
410 IF M<13 GOTO 450
420 LET M=1: LET G=G+1: GOSUB 100
450 FOR I=0 TO B
460 LET Q(I)=I: NEXT I
470 GOSUB 20: FOR I=0 TO B
480 PRINT TAB INT(P(I)+.5)+27
490 ON Q(I)+500
500 PRINT 'I' !E!: GOTO 510
501 PRINT 'Э' !E!: GOTO 510
502 PRINT 'Ф' !E!: GOTO 510
503 PRINT 'И' !E!: GOTO 510
510 NEXT I: PRINT
520 NEXT C: PRINT
530 STOP
999 END
:
```

Рис. 15. Программа БИОРИТМЫ.


```

RUN
ДАТА.МЕСЯЦ, ГОД РОЖДЕНИЯ      16.04,1952
ДАТА.МЕСЯЦ, ГОД НАЧАЛА ГРАФИКА 20.11,1979
ДАТА.МЕСЯЦ, ГОД КОНЦА ГРАФИКА  31.12,1979

20.11          Э      I      И      Ф
21.11          Э      И      Ф
22.11          I      И      Э      Ф
23.11          И      I      Э      Ф
24.11          И      I      Э      Ф
25.11          И      I      Ф      Э
26.11          И      I      Ф      Э
27.11          И      Ф      I      Э
28.11          И      Ф      I      Э
29.11          И      Ф      I      Э
30.11          И      Ф      I      Э
1.12          ИФ      I      Э
2.32          ФИ      I      Э
3.12          Ф      И      I      Э
4.12          Ф      И      I      Э
5.12          Ф      И      Э
6.12          ИФ      Э      I
7.12          Э      И      Ф      I
8.12          Э      И      Ф      I
9.12          Э      И      Ф      I
10.12         Э      И      Ф      I
11.12         Э      И      Ф      I
12.12         Э      И      Ф      I
13.12         Э      И      Ф      I
14.12         Э      И      Ф      I
15.12         Э      И      Ф      I
16.12         Э      И      Ф      I
17.12         Э      И      Ф      I
18.12         Э      И      Ф      I
19.12         Э      И      Ф      I
20.12         Э      И      Ф      I
21.12         Э      И      Ф      I
22.12         Э      И      Ф      I
23.12         Э      И      Ф      I
24.12         Э      И      Ф      I
25.12         Э      И      Ф      I
26.12         Э      И      Ф      I
27.12         Э      И      Ф      I
28.12         Э      И      Ф      I
29.12         Э      И      Ф      I
30.12         Э      И      Ф      I
31.12         Э      И      Ф      I

ОСТАНОВ В СТРОКЕ      530
:
```

Рис. 16. Пример работы программы БИОРИТМЫ.

Э – «эмоционального» (период 28 дней);

И – «интеллектуального» (период 33 дня).

При желании число ритмов может быть расширено путём изменения строки 320 и добавления строк 504–509 и 370.

Идея алгоритма заимствована из [4].

5.2.2. В качестве упражнения на закрепление Ваших знаний о Бэйсике, попробуйте разобраться в этой программе и улучшить её быстродействие.

5.2.3. Пример протокола работы программы приведён на [рис. 16](#).

```

LIST 1,499

5 PRINT
10 INPUT 'N ? 'N1: LET N1=N1-1           ← N – порядок матрицы
11 DIM Y(N1,N1+N1+1)
12 LET N2=2*N1+1
13 LET N3=N1+1
30 FOR I=0 TO N1
40 FOR J=0 TO N1
50 READ Y(I,J)
60 NEXT J
70 NEXT I
79 PRINT: PRINT
80 FOR I=0 TO N1
90 FOR J=N3 TO N2
100 IF I=J-N3 THEN LET Y(I,J)=1: GOTO 120
110 LET Y(I,J)=0
120 NEXT J
130 NEXT I
140 GOTO 330
150 LET I1=K1
151 IF Y(I1,K1)<>0 THEN LET K2=I1: GOTO 190
152 FOR I1=K1 TO N1
160 IF Y(I1,K1)<>0 THEN LET K2=I1: GOTO I90
170 NEXT I1
180 IF I1=N1 GOTO 839
190 FOR J1=K1 TO N2
200 LET B1=Y(K1,J1)
210 LET Y(K1,J1)=Y(K2,J1)
220 LET Y(K2,J1)=B1
230 NEXT J1
240 FOR J2=K1+1 TO N2
250 LET Y(K1,J2)=Y(K1,J2)/Y(K1,K1)
260 NEXT J2
270 FOR I3=K1+1 TO N1
280 FOR J3=K1+1 TO N2
290 LET Y(I3,J3)=Y(I3,J3)-Y(K1,J3)*Y(I3,K1)
300 NEXT J3
310 NEXT I3
320 RETURN
330 LET K1=0
340 GOSUB 150
350 LET K2=K1
360 IF K2=N1-1 GOTO 390
370 LET K1=K1+1
380 GOTO 340
390 LET B3=1
400 FOR I=0 TO N1
420 LET B3=B3*Y(I,I)
440 NEXT I
450 IF B3=0 GOTO 839
460 FOR I=N3 TO N2
470 LET Y(N1,I)=Y(N1,I)/Y(N1,N1)
480 NEXT I
490 LET K2=N1-1
:

```

Рис. 17 (начало). Программа ОБРАЩЕНИЕ МАТРИЦЫ.

```

LIST 500,999

500 FOR I=0 TO K2
510 FOR J=N3 TO N2
520 LET Y(I,J)=Y(I,J)-Y(I,K2+1)*Y(K2+1,J)
530 NEXT J
540 NEXT I
550 LET K2=K2-1
560 IF K2>=0 GOTO 500
565 PRINT 'ОБРАТНАЯ МАТРИЦА   Y(I,J) '
570 LET K2=0
580 LET K1=K2
590 LET K2=K1+3
600 IF N1<K2 THEN LET K2=N1
601 PRINT: PRINT
610 PRINT 'Y(I,J)';
620 FOR I=K1+1 TO K2+1
621 PRINT TAB 12+15*(I-INT((I-1)/4)*4-1)
624 PRINT !2.0!
625 PRINT I;
630 NEXT I
640 PRINT
650 FOR I=1 TO N1+1
655 PRINT !2.0!
660 PRINT I;
665 PRINT TAB 8
670 FOR J=K1 TO K2
679 PRINT !F1.5!
680 PRINT Y(I-1,J+N1+1);;;
690 NEXT J
700 PRINT
710 NEXT I
720 IF J>=N1 GOTO 770
730 IF N1-J>4 THEN LET K1=K2+1: GOTO 590
740 LET K1=J+1
750 LET K2=N1
760 PRINT: GOTO 610
768 LET B3=ABS(B3)
770 LET B4=ABS(Y(0,N3))
780 FOR I=0 TO N1
790 FOR J=0 TO N1
799 LET B5=ABS(Y(I,J+N3))
800 IF B4<=B5 THEN LET B4=B5: GOTO 810
810 NEXT J
820 NEXT I
821 LET B4=B4/B3: IF B4<1E11 GOTO 840
822 PRINT 'СИСТЕМА ПЛОХО ОБУСЛОВЛЕНА'
823 PRINT 'ТРЕБУЕТСЯ УТОЧНЕНИЕ МАТРИЦЫ Y(I,J)': GOTO 840
839 PRINT 'ОПРЕДЕЛИТЕЛЬ = 0'
901 DATA 1,2,3,4,5
902 DATA 2,3,4,5,1
903 DATA 3,4,5,1,2
904 DATA 4,5,1,2,3
905 DATA 5,1,2,3,4
999 END
:

```

} элементы матрицы

Рис. 17 (окончание). Программа ОБРАЩЕНИЕ МАТРИЦЫ.

```

RUN
N ? 5
ОБРАТНАЯ МАТРИЦА   Y(I, J)
Y(I, J)           1           2           3           4
1      -1.86667E-01    1.33333E-02    1.33333E-02    1.33333E-02
2       1.33333E-02    1.33333E-02    1.33333E-02    2.13333E-01
3       1.33333E-02    1.33333E-02    2.13333E-01   -1.86667E-01
4       1.33333E-02    2.13333E-01   -1.86667E-01    1.33333E-02
5       2.13333E-01   -1.86667E-01    1.33333E-02    1.33333E-02
Y(I, J)           5
1       2.13333E-01
2      -1.86667E-01
3       1.33333E-02
4       1.33333E-02
5       1.33333E-02
ОСТАНОВ В СТРОКЕ   999
:50 Y(I, J)=Y(I, J+N3)
:GOTO 12
ОБРАТНАЯ МАТРИЦА   Y(I, J)
Y(I, J)           1           2           3           4
1       1.00000      2.00000      3.00000      4.00000
2       2.00000      3.00000      4.00000      5.00000
3       3.00000      4.00000      5.00000      1.00000
4       4.00000      5.00000      1.00000      2.00000
5       5.00000      1.00000      2.00000      3.00000
Y(I, J)           5
1       5.00000
2       1.00000
3       2.00000
4       3.00000
5       4.00000
ОСТАНОВ В СТРОКЕ   999
:

```

Рис. 18. Пример протокола работы программы ОБРАЩЕНИЕ МАТРИЦЫ.

5.3. Обращение матрицы

5.3.1. На [рис. 17](#) приведена программа обращения матрицы n -го порядка методом Гаусса с выбором главного элемента.

Элементы матрицы задаются в блоке данных (строки 900–998). Порядок матрицы вводится оператором **INPUT** (строка 10).

5.3.2. Пример протокола работы программы приведён на [рис. 18](#). Для проверки обращения обратная матрица снова обращена. Перепись элементов произведена путём модификации строки 50, после чего программа запущена со строки 12, т.к. при запуске по **RUN** оператор **DIM** обнулil бы массив **Y**.

5.4. Информационно-справочная система

5.4.1. Приведённая на [рис. 19](#) простейшая программа **СПРАВОЧНАЯ СИСТЕМА** иллюстрирует применение НМЛ для хранения сегментов программ и данных.

```
LIST 1,699
5 COM N1(0),R(0)
10 PRINT: PRINT TAB 15 'СПРАВОЧНАЯ СИСТЕМА СРЕДНИХ ОЦЕНОК'
30 PRINT: LET N=0: REWIND
50 INPUT 'РЕЖИМ ? (0 - КОНЕЦ РАБОТЫ, 1 - ПЕЧАТЬ, 2 - ВВОД ДАННЫХ)'R
60 PRINT
70 IF R=0 THEN PRINT 'КОНЕЦ РАБОТЫ': PRINT: STOP
80 INPUT 'НОМЕР ГРУППЫ ? 'N1: PRINT
90 IF N1=N GOTO 180
100 IF N1<N THEN REWIND
110 GOSUB 5000
120 IF END THEN PRINT 'НЕТ НА ЛЕНТЕ': GOTO 30
130 RESTORE: CLEAR D: READ L
140 DIM K(L),S(L)
150 DATA LOAD N,K(),S()
160 IF END GOTO 200
170 IF N1<>N GOTO 200
180 IF R=1 GOTO 300
190 GOTO 500
200 PRINT
210 PRINT 'СВОЙ СИСТЕМЫ'
220 PRINT
230 GOTO 30
300 PRINT
310 PRINT '                ГРУППА';
320 GOSUB 6000
325 PRINT
330 PRINT '                КОЛИЧЕСТВО  СРЕДНЯЯ'
340 PRINT '                ОЦЕНОК      ОЦЕНКА'
350 PRINT
360 LET K=0: LET S=0
370 FOR I=1 TO L: GOSUB 7000
380 PRINT TAB 22 !4.0! K(I) TAB 33 !3.2! S(I)/K(I)
390 LET K=K+K(I): LET S=S+S(I)
400 NEXT I
410 PRINT
420 PRINT 'СРЕДНИЕ ПО ГРУППЕ':
430 PRINT TAB 22 !3.1! K/L TAB 33 !3.2! S/K
440 PRINT
450 GOTO 50
500 PRINT
510 PRINT '                ВВОД ОЦЕНОК': PRINT
520 PRINT 'НОМЕР      ФАМИЛИЯ                ОЦЕНКА'
530 FOR I=1 TO L
540 PRINT !3.0! I TAB 9
550 GOSUB 7000
560 PRINT TAB 28
570 INPUT '? 'S: IF S<=0 THEN 590
580 LET K(I)=K(I)+1: LET S(I)=S(I)+S
590 NEXT I
600 REWIND
610 GOSUB 4000: IF END GOTO 200
620 DATA SAVE N,K(),S()
625 PRINT
630 GOTO 50
:
```

Рис.19 (начало). Программа СПРАВОЧНАЯ СИСТЕМА.


```
LIST 4000,7999
4000 ON 4000+N
4001 DATA LOAD 'ГР 1': RETURN
4002 DATA LOAD 'ГР 2': RETURN
4003 DATA LOAD 'ГР 3': RETURN
4004 DATA LOAD 'ГР 4': RETURN
4005 DATA LOAD 'ГР 5': RETURN
4900 CLEAR S: GOTO 200
5000 ON 5000+N1*10
5010 LOAD 'ГР 1' 6000,7999
5011 RETURN
5020 LOAD 'ГР 2' 6000,7999
5021 RETURN
5030 LOAD 'ГР 3' 6000,7999
5031 RETURN
5040 LOAD 'ГР 4' 6000,7999
5041 RETURN
5050 LOAD 'ГР 5' 6000,7999
5051 RETURN
5900 PRINT 'ОШИБОЧНЫЙ НОМЕР ГРУППЫ'
5910 CLEAR S: PRINT
5920 GOTO 80
:
```

Рис. 19 (окончание). Программа СПРАВОЧНАЯ СИСТЕМА.

```
LIST 6000,7999
6000 DATA 11
6010 PRINT 'ГРУППА 1': RETURN
7000 ON 7000+I
7001 PRINT 'ИВАНОВ - 1';: RETURN
7002 PRINT 'ИВАНОВ - 2';: RETURN
7003 PRINT 'ИВАНОВ - 3';: RETURN
7004 PRINT 'ИВАНОВ - 4';: RETURN
7005 PRINT 'ИВАНОВ - 5';: RETURN
7006 PRINT 'ИВАНОВ - 6';: RETURN
7007 PRINT 'ИВАНОВ - 7';: RETURN
7008 PRINT 'ИВАНОВ - 8';: RETURN
7009 PRINT 'ИВАНОВ - 9';: RETURN
7010 PRINT 'ИВАНОВ - 10';: RETURN
7011 PRINT 'ИВАНОВ - 11';: RETURN
7998 PRINT 'НЕТ ФАМИЛИИ';: RETURN
7999 END
:
```

Рис.20. Пример программного файла СПРАВОЧНОЙ СИСТЕМЫ.

5.4.2. Система предназначена для учёта текущей успеваемости студентов нескольких групп. Сведения о каждой группе представлены на МЛ двумя файлами – программным и файлом данных.

Программный файл, пример которого приведён на рис. 20, содержит сведения о количестве студентов в группе (строка 6000), наименование группы и фамилии студентов. В файле данных записаны номер группы N , число оценок $K(I)$; и сумма оценок $S(I)$ для каждого студента группы.

Структура файлов на МЛ для трёх групп студентов приведена на рис. 21.



Рис.21. Структура МЛ для трёх групп студентов.

```

PRINT OPEN
Р ФОРМИРОВАНИЕ МАГНИТНОЙ ЛЕНТЫ ДЛЯ СПРАВОЧНОЙ СИСТЕМЫ
:LIST 1,999

5 COM N(0),K(0),L(0)
10 REWIND
20 INPUT 'КОЛИЧЕСТВО ГРУПП ? 'N
30 LET I=1
40 PRINT 'ВВЕДИТЕ ТЕКСТ СТРОК 6000 - 7999 ДЛЯ ГРУППЫ '!2.0! I','
50 PRINT 'ЗАТЕМ ПОДАЙТЕ КОМАНДУ GOTO 60': STOP
60 RESTORE
70 READ L
80 CLEAR D
90 DIM K(L),S(L)
100 ON 100+10*I
110 SAVE 'ГР 1' 6000,7999
114 DATA SAVE OPEN 'ГР 1'
118 GOTO 300
120 SAVE 'ГР 2' 6000,7999
124 DATA SAVE OPEN 'ГР 2'
128 GOTO 300
130 SAVE 'ГР 3' 6000,7999
134 DATA SAVE OPEN 'ГР 3'
138 GOTO 300
140 SAVE 'ГР 4' 6000,7999
144 DATA SAVE OPEN 'ГР 4'
148 GOTO 300
150 SAVE 'ГР 5' 6000,7999
154 DATA SAVE OPEN 'ГР 5'
158 GOTO 300
300 SAVE END
310 REWIND
320 ON 320+I
321 DATA LOAD 'ГР 1': GOTO 350
322 DATA LOAD 'ГР 2': GOTO 350
323 DATA LOAD 'ГР 3': GOTO 350
324 DATA LOAD 'ГР 4': GOTO 350
325 DATA LOAD 'ГР 5': GOTO 350
350 IF END THEN 500
360 DATA SAVE I,K(),S()
370 DATA SAVE END
380 LET I=I+1: IF I<=N THEN 40
390 PRINT: SAVE END
400 PRINT 'ЛЕНТА СФОРМИРОВАНА'
410 PRINT
420 STOP
500 PRINT 'СМЕНИТЕ ЛЕНТУ'
510 RUN
:

```

Рис.22. Программа ФОРМИРОВАНИЕ МЛ.

```

LIST 1,38
1 GOTO 110
2 LET J=V+G*S: LET I=A-S*(V+G*S/2): IF K=0 THEN RETURN
3 LET Q=1-K*S/M: LET J=J+Z*LOG(Q): LET I=I+Z*S+Q*LOG(Q)*Z/K*M: RETURN
4 LET L=L+S: LET T=T-S:LET M=M-S*(K+F): LET A=I: LET V=J: RETURN
5 IF L=B THEN 95
6 PRINT !6.0! L; INT(A); (A-INT(A))*5280; 5280*V; M-N; !5.2!
7 INPUT '?'K: LET T=10: IF K>=8 THEN IF K<=200 GOTO 9
8 IF K<>0 THEN PRINT 'СТОЛЬКО ГОРЮЧЕГО РАСХОДОВАТЬ НЕЛЬЗЯ!';: GOTO 7
9 LET Q=RND(3): IF Q>.98 GOTO 23
10 LET Z=1,8*(1-H)
11 IF M-N<1E-3 GOTO 27
12 IF T<1E-3 GOTO 5
13 LET S=T: IF N+S*(K+F)>M THEN LET S=(M-N)/(K+F)
14 GOSUB 2: IF I<=0 GOTO 17
15 IF V>0 THEN IF J<0 GOTO 19
16 GOSUB 4:GOTO 11
17 IF S<5E-3 GOTO 28
18 LET S=2*A/(V+SQR(V*V+2*A*(G-(Z*K/M)))): GOSUB 2: GOSUB 4: GOTO 17
19 LET W=(1-M*G/(Z*K))/2: LET S=M*V/(Z*K*(W+SQR(W*W+V/2)))+5E-2: GOSUB 2
20 IF I<=0 GOTO 17
21 GOSUB 4: IF J<0 THEN IF V>0 GOTO I9
22 GOTO 11
23 IF Q>.998 THEN PRINT 'ЛУННЫЙ МОДУЛЬ УНИЧТОЖЕН МЕТЕОРИТОМ': GOTO 52
24 PRINT 'УДАР МИКРОМЕТЕОРИТА,';: LET Q1=.5+RND(3)
25 IF Q<.99 THEN LET F=F+Q1: PRINT 'УТЕЧКА ГОРЮЧЕГО'F' ФУНТОВ/С': GOTO 10
26 PRINT 'ТЯГА ДВИГАТЕЛЯ УПАЛА НА'Q1'%': LET H=H+.01*Q1: GOTO 10
27 PRINT 'ГОРЮЧЕЕ КОНЧИЛОСЬ НА'L'С': LET S=(SQR(V*V+2*A*G)-V)/G: LET V=V+G*S: LET L=L+S
28 PRINT 'ВЫ НА ЛУНЕ НА'L' С,';: LET W=5280*V: PRINT 'СКОРОСТЬ СТОЛКНОВЕНИЯ'W' ФУТОВ/С'
29 LET R=M-N: PRINT 'ГОРЮЧЕГО ОСТАЛОСЬ'R' ФУНТОВ': GOTO 38
30 PRINT ' ПОСАДКА,';: LET Y=60-.08*(L-160)-1.5*W*W+15*F+25*H: IF R<550 GOTO 35
31 LET Y=Y+(.01*(R-550))-3: ON 32+X
32 PRINT 'ВЫ ПРЕКРАСНЫЙ СНАЙПЕР': RETURN
33 PRINT 'ВЫ ОТЛИЧНЫЙ НАЕЗДНИК': RETURN
34 PRINT 'НЕ ОТЛИЧНАЯ, НО, В ОБЩЕМ-ТО, НЕПЛОХО': RETURN
35 PRINT 'НО ГОРЮЧЕГО ДЛЯ ВЗЛЕТА ';: LET Y=Y-2*(.01*(550-R))-3
36 IF R<400 THEN PRINT 'НЕ ХВАТИТ': LET Y=Y-.1*(.01*(400-R))-4: RETURN
37 PRINT 'МАЛОВАТО': RETURN
38 IF W>-2.85 GOTO 47
:
```

Рис.23 (начало). Программа ПОСАДКА НА ЛУНУ.

```
LIST 39,76
39 IF D+E+O+P+C=0 THEN PRINT 'ВЫ СНЯТЫ С ПОЛЕТОВ ЗА ПОЛЬЗОВАНИЕ ШПАРГАЛКОЙ': GOTO 44
40 LET Z4=Z4+1: LET U=0: IF W>.89 THEN 45
41 PRINT 'ВИРТУОЗНАЯ';: LET X=0: GOSUB 30
42 PRINT 'ВАШЕ ЖАЛОВАНИЕ УВЕЛИЧЕНО НА'Y' ДОЛЛ./НЕД.': IF Z4=1 GOTO 52
43 PRINT 'ТРЕНИРОВКА ОКОНЧЕНА - ВЫ УЖЕ ХОРОШО ЛЕТАЕТЕ'
44 PRINT: PRINT 'КТО СЛЕДУЮЩИЙ?': PRINT: GOTO 96
45 PRINT 'БЕЗУПРЕЧНАЯ';: LET X=1: GOSUB 30: LET Y=.6*Y: IF Z4=1 GOTO 52
46 GOTO 42
47 IF W>9.7 GOTO 56
48 PRINT 'ХОРОШАЯ';: LET X=2: GOSUB 30: LET U=0: LET D=D+1: ON 52-D
49 PRINT 'ВАШИ РЕЗУЛЬТАТЫ УЛУЧШАЮТСЯ НЕДОСТАТОЧНО БЫСТРО': GOTO 44
50 PRINT 'ВСЕ ЖЕ НЕ СЛЕДУЕТ УСПОКАИВАТЬСЯ НА ДОСТИГНУТОМ': GOTO 52
51 IF E+O+P+C=0 THEN PRINT 'ВЫ ПОДОЗРЕВАЕТЕСЬ В ПОЛЬЗОВАНИИ ШПАРГАЛКОЙ'
52 PRINT: PRINT 'НАЧНЕТЕ С ШАГА 0 (0) ИЛИ С ШАГА' !3.0! 'В' (1)';
53 INPUT '?F7: IF F7=0 GOTO 93
54 IF F8=0 GOTO 93
55 LET L=L1: LET A=A1: LET V=V1: LET M=M1: LET F=F1: LET H=H1: GOTO 94
56 IF W>27.3 THEN 65
57 PRINT 'ЭТО ПОХОЖЕ НА ПРЫЖОК КОЗЛЯ, НО ПРИМИТЕ ПОЗДРАВЛЕНИЯ - ВЫ ВНИЗУ'
58 LET E=E+1: IF E>1 THEN ON 59+E
59 IF U>1 THEN PRINT 'ПРОДОЛЖАЙТЕ ЭКСПЕРИМЕНТИРОВАТЬ В ТОМ ЖЕ НАПРАВЛЕНИИ': LET U=1: GOTO 52
60 PRINT 'ВЫ УХУДИЛИ СВОЙ ГРАФИК ПОЛЕТА': LET U=1: GOTO 52
61 PRINT 'ХОРОШО БЫ ПОАККУРАТНЕЕ...': LET U=1: GOTO 52
62 PRINT 'ВАШЕ ЛИХАЧЕСТВО - ДУРНОЙ ПРИМЕР ДЛЯ МОЛОДЫХ КУРСАНТОВ': LET U=1: GOTO 52
63 PRINT 'ВЫ УПОРНО ПРОДОЛЖАЕТЕ ПУТАТЬ АСТРОНАВТА С КОВБОЕМ'
64 PRINT !3.0! Z4*D+E+O+P+C'-Я ПОПЫТКА ОКОНЧЕНА': GOTO 44
65 IF W>47 GOTO 70
66 PRINT 'ЛЕГКИЕ ПОВРЕЖДЕНИЯ КОРПУСА И НАРУЖНЫХ КОНСТУКЦИЙ': LET O=O+1
67 LET Y1=Y1+17.24+.85*(W-27.3): PRINT 'NASA БУДЕТ ВЫЧИТАТЬ'Y1' ДОЛЛ. С ВАШЕГО СЧЕТА ЕЖЕНЕДЕЛЬНО'
68 LET U=2: IF O<5 GOTO 52
69 PRINT 'ВАШ СЧЕТ В БАНКЕ ИСЧЕРПАН': GOTO 64
70 IF W>85 GOTO 82
71 PRINT 'ТЯЖЕЛЫЕ РАЗРУШЕНИЯ, ВЫ НИКОГДА БОЛЬШЕ НЕ УВИДИТЕ ЗЕМЛЮ'
72 LET P=P+1: IF P>1 THEN LET U=3: ON 75+P
73 IF U<3 THEN PRINT 'ВАШИ ПОПРАВКИ ГРАФИКА ПОЛЕТА НЕУДАЧНЫ': GOTO 76
74 IF C>0 THEN PRINT 'УЖЕ ЛУЧШЕ,';
75 PRINT 'НАДЕЕМСЯ, СЛЕДУЮЩАЯ ПОПЫТКА БУДЕТ УДАЧНЕЕ'
76 LET U=3: GOTO 52
:
```

Рис.23 (продолжение). Программа ПОСАДКА НА ЛУНУ.

```
LIST 77,7999
77 PRINT 'ПОКАЛЕЧИЛИСЬ ВТОРИЧНО': GOTO 52
78 PRINT 'NASA ОБУЧАЕТ ПИЛОТОВ НЕ ДЛЯ ТОГО, ЧТОБЫ ПЛАТИТЬ ИМ ЗА ИНВАЛИДНОСТЬ': GOTO 52
79 PRINT 'ВАМ БЫ ЛУЧШЕ УВОЛИТЬСЯ ИЗ NASA': GOTO 52
80 PRINT 'НЕЛЬЗЯ ЖЕ ДЛЯ ВАС ОБЛОЖИТЬ ВСЮ ЛУНУ ПОДУШКАМИ... ': GOTO 52
81 PRINT 'ЭТА РАБОТА НЕ ДЛЯ ВАС - НА ЛУНЕ НЕТ ВРАЧЕЙ': GOTO 44
82 PRINT 'ВЫ ОБРАЗОВАЛИ НОВЫЙ КРАТЕР НА ЛУНЕ ГЛУБИНОЙ'.19*W' ФУТОВ'
83 LET C=C+1: ON 91-C
84 PRINT 'NASA НЕ БУДЕТ ТРАТИТЬ ДЕНЬГИ НА ВАШЕ ОБУЧЕНИЕ': GOTO 64
85 PRINT 'СОВЕТУЕМ ПРИМЕНИТЬ СВОИ СИЛЫ В СФЕРЕ ОБСЛУЖИВАНИЯ': GOTO 92
86 PRINT 'ВАМ ЕЩЕ НЕ НАДОЕЛО УМИРАТЬ?': GOTO 92
87 PRINT 'ВОТ ВЫ И ОПЯТЬ МЕРТВЫ... ': GOTO 92
88 PRINT 'СМЕРТЬ В ТРЕТИЙ РАЗ...': GOTO 92
89 PRINT 'ВТОРИЧНАЯ СМЕРТЬ': GOTO 92
90 IF U=4 THEN PRINT 'НУ ЧТО ЖЕ, ЭТО ОБЫЧНЫЙ ИСХОД ПЕРВОЙ ПОПЫТКИ': GOTO 92
91 PRINT 'МЕНЯЙТЕ ГРАФИК ПОЛЕТА В ДРУГОМ НАПРАВЛЕНИИ'
92 LET U=4: GOTO 52
93 LET L=0:LET T=0:LET A=120:LET S=1:LET V=1:LET F8=0:LET M=32500:LET F=0:LET N=16500:LET G=1E-3:LET H=0
94 INPUT 'ЗАПОМНИТЬ ШАГ? (ДА - ВРЕМЯ, НЕТ - НУЛЬ) 'B: LET B=10*INT(.1*B+.5): GOTO 5
95 LET L1=L: LET A1=A: LET V1=V: LET M1=M: LET F8=1: LET F1=F: LET H1=H: GOTO 6
96 INPUT 'ИНСТРУКЦИЯ? (ДА - 1, НЕТ - 0) 'B
97 IF B>0 GOTO 99
98 LET Z4=0: LET D=0: LET E=0: LET O=0: LET P=0: LET C=0: LET Y1=0: LET U=4: GOTO 93
99 PRINT: PRINT 'ВАМ НУЖНО ПОСАДИТЬ ЛУННЫЙ МОДУЛЬ С ПОЛЕЗНЫМ ВЕСОМ 16500 ФУНТОВ'
100 PRINT 'НА РУЧНОМ УПРАВЛЕНИИ ПО ПОКАЗАНИЯМ ПРИБОРОВ';
101 CMD 1406 0310: PRINT
102 PRINT '1-Й СТОЛБЕЦ - ВРЕМЯ В СЕКУНДАХ'
103 PRINT '2-Й И 3-Й - РАССТОЯНИЕ ДО ПОВЕРХНОСТИ ЛУНЫ В МИЛЯХ ПЛЮС ФУТАХ'
104 PRINT '4-Й - СКОРОСТЬ В ФУТАХ/С'
105 PRINT '5-Й - ЗАПАС ГОРЮЧЕГО В ФУНТАХ.'
106 PRINT 'В ШЕСТОМ СТОЛБЦЕ ВЫ ДОЛЖНЫ ЗАДАТЬ РАСХОД ГОРЮЧЕГО - НУЛЬ ИЛИ'
107 PRINT 'ОТ ВОСЬМИ ДО ДВУХСОТ ФУНТОВ В СЕКУНДУ ВКЛЮЧИТЕЛЬНО'
108 PRINT: PRINT 'НАЧИНАЙТЕ ПРОЦЕДУРУ ПОСАДКИ': PRINT: GOTO 98
110 INPUT 'ТАБЛИЦА РЕКОРДОВ? (ДА - 1, НЕТ - 0) 'B
120 IF B=0 GOTO 96
130 PRINT: PRINT 'НАДБАВКА К ЖАЛОВАНИЮ - 61.96$ В НЕД. (БОЛОТИН В.Н.)'
140 PRINT 'СКОРОСТЬ МЯГКОЙ ПОСАДКИ - 0.02 ФУТ/С (СЕМЕНЦОВ Э.А.)'
150 PRINT 'ПРОДОЛЖИТЕЛЬНОСТЬ ПОЛЕТА - 644.34 С (ИПАТОВ М.А.)'
160 PRINT: GOTO 96
1000 LET B=RND(0): RUN
7999 END
```

Рис.23 (окончание). Программа ПОСАДКА НА ЛУНУ.

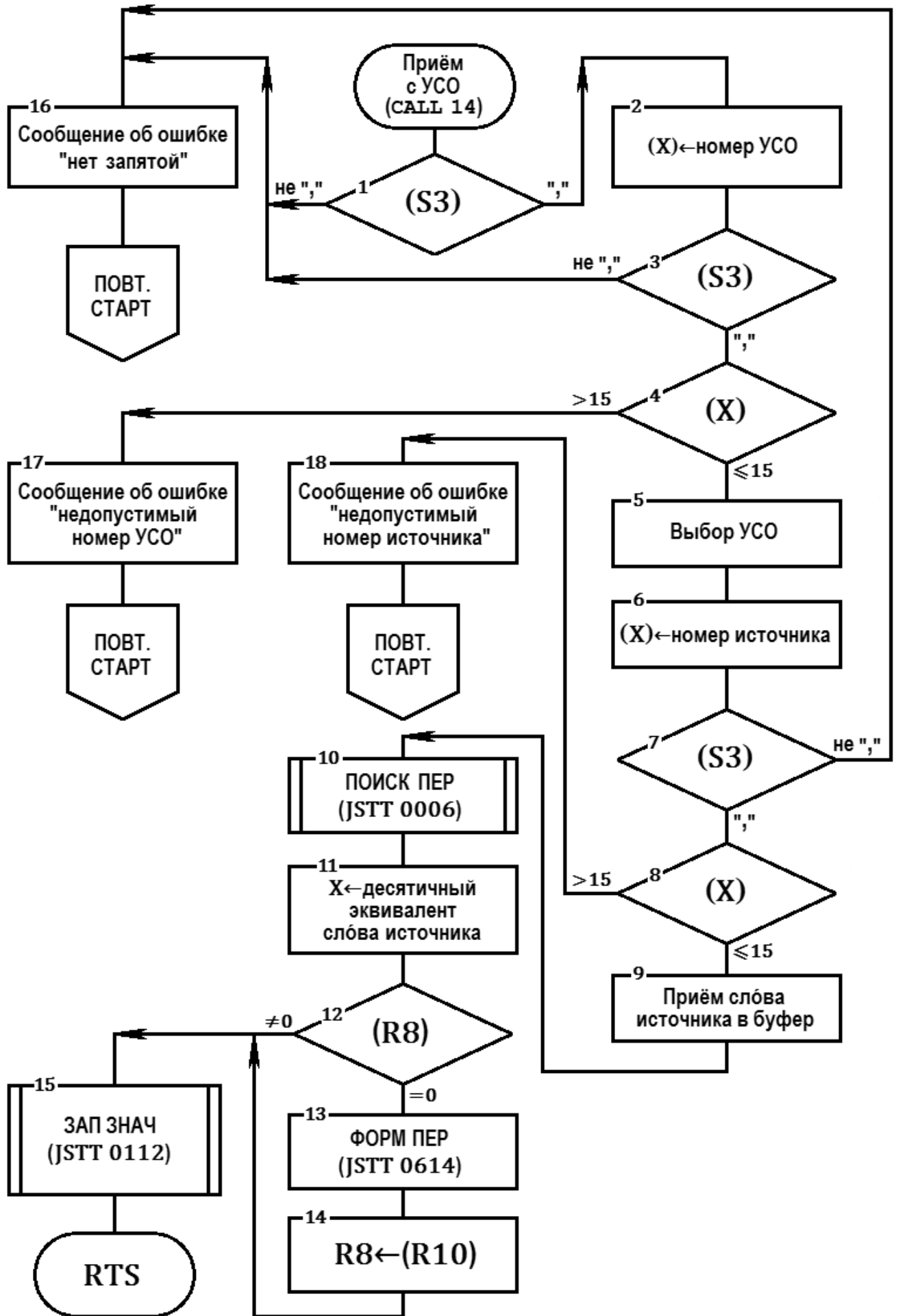


Рис.24. Алгоритм приёма с УСО.

Файлы могут быть сформированы как вручную (в непосредственном режиме), так и полуавтоматически. Вариант программы полуавтоматического формирования МЛ приведён на [рис. 22](#).

Обратите внимание на принципиальную невозможность организации цикла по переменной **I** в этой программе операторами **FOR–NEXT**, поскольку при изменении размеров программы пользователя при вводе текста строк адрес управляющей переменной в ОЗУ может измениться, что приведёт к неправильной работе оператора **NEXT**.

5.4.3. Для ускорения поиска файлов на МЛ можно включать перемотку на заданное время с помощью оператора **CMD**. Например, в программе **СПРАВОЧНАЯ СИСТЕМА** строка 600 может быть записана в виде:

```
600 CMD 1200, 1304, 1515, 0412 1402, 0412 1200
```

5.5. Посадка на Луну

5.5.1. Эта игровая программа является незначительной переработкой Бэйсик-программы для машин серии СМ неизвестного нам автора.

5.5.2. Текст программы приведён на [рис. 23](#).

5.6. Ввод-вывод на УСО

5.6.1. В качестве примера внешних подпрограмм, использующих внутренние подпрограммы интерпретатора, рассмотрим подпрограммы обслуживания устройства связи с объектами 15КС-16 (УСО).

5.6.2. К каждому УСО может быть подключено 16 приёмников и 16 источников данных – двухбайтовых слов. Источники и приёмники адресуются их номерами от нуля до пятнадцати. К ДЗ-28 может быть подключено несколько УСО, отличающихся номерами (от 0 до 15).

Будем считать, что передаваемые данные являются целыми двоичными числами в прямом коде со знаком, причём знак записан в старшем бите старшего байта двухбайтового слова.

Нашей задачей является разработка внешних подпрограмм, обеспечивающих ввод и вывод данных на УСО.

5.6.3. Приём данных с УСО будет осуществлять внешняя программа, скажем, с номером 14. Положим, что вызов этой подпрограммы будет производиться предложением **CALL 14, <номер УСО>, <номер источника>, <переменная>**. Например, для того чтобы принять данные с источника 13 УСО 5 и присвоить принятое значение переменной **T5**, нужно будет записать предложение **CALL 14, 5, 13, T5**.

Принципиальный алгоритм приёма с УСО приведён на [рис. 24](#), а вариант его реализации – на [рис. 25](#).

Выбор УСО в подпрограмме осуществляется командой с кодами 0409 02 <номер УСО>, а приём слова от источника – командой 1500 01 <номер источника>.

5.6.4. Аналогично строится подпрограмма вывода, например, с номером 325, вызываемая предложением **CALL 14, <номер УСО>, <номер приёмника>, <выраже-**

Первый блок:

00000	0000	}	номер подпрограммы (14)
00001	0104		
00002	0010	}	контрольная сумма (2560 ₁₀)
00003	0000		
00004	0000	}	количество шагов (195 ₁₀)
00005	1203		
00006	0000		
00007	0512		

Второй блок:

```

00000 1009 0212  BNS #0212,S3 00004
00002 1403 0003  BR 00006
00004 1403 0007  BR 00012
00006 1003 0200  ATOI 0200           ; чтение нóмера УСО
00008 1009 0212  BNS #0212,S3 00012
00010 1403 0101  BR 00028
00012 1300 0000  MOV #0000,S00       ; формирование
00014 1301 0607  MOV #0607,S01       ; адреса сообщения
00016 1105 1015  MOV R15 ,R10       ; об ошибке
00018 0413 0810  ABS R10
00020 1100 0810  ADD R08,R10
00022 1013 0606  JSTT 0606           ; ВЫВОД СИМВ
00024 1013 0304  JSTT 0304           ; "В СТРОКЕ"
00026 0407 0408  JMM 0408           ; на ПОВТ. СТАРТ
00028 1306 0000  MOV #0000,S06
00030 1307 0015  MOV #0015,S07
00032 0413 1208  MOVH X,R08
00034 0510      BPER 00037
00035 1403 0007  BR 00043
00037 1300 0000  MOV #0000,S00
00039 1301 0704  MOV #0704,S01
00041 1402 0110  BR 00016
00043 1410 1108  BGE R11,R08 00047
00045 1402 0009  BR 00037
00047 1302 0409  MOV #0409,S02       ; формирование
00049 1303 0200  MOV #0200,S03       ; команды выбора
00051 1100 0809  ADD R08,R09
00053 0413 0209  CMD R09           ; выбор УСО
00055 1003 0200  ATOI 0200       ; чтение номера
00057 1009 0212  BNS #0212,S3 00061 ; источника
00059 1403 0003  BR 00063
00061 1402 0302  BR 00012
00063 0413 1208  MOVH X,R08
00065 0510      BPER 00068
00066 1403 0007  BR 00074
00068 1300 0000  MOV #0000,S00
00070 1301 0812  MOV #0812,S01
00072 1402 0309  BR 00016
00074 1410 1109  BGE R11,R09 00078

```

Рис. 25 (начало). Внешняя подпрограмма приёма с УСО.

00076	1402	0009	BR 00068	
00078	1013	1114	JSTT 1114	; ОБЪЁМ ОЗУ
00080	1304	0312	MOV #0312,S04	; формирование
00082	1305	0708	MOV #0708,S05	; адреса
00084	1111	0904	OR S09,S04	; буфера
00086	1302	1500	MOV #1500,S02	; формирование
00088	1303	0100	MOV #0100,S03	; команды
00090	1101	0809	SUB R08,R09	; приёма
00092	1104	0212	MOV R02,R12	; к-во байтов
00094	0413	0209	CMD R09	; приём с УСО
00096	0905	0810	MOV @R10,R08	
00098	1012	0813	MOV R08,-(R13)	; запись слóва в стек
00100	1013	0006	JSTT 0006	; ПОИСК ПЕР
00102	1015	0413	MOV (R13)+,R04	
00104	0413	0404	MOVD R04,X	; десят. эквивалент
00106	1103	0408	BSAZ R08 00111	
00108	1013	0112	JSTT 0112	; ЗАП ЗНАЧ
00110	0511		RTS	
00111	1013	0614	JSTT 0614	; ФОРМ ПЕР
00113	1105	0810	MOV R10,R08	
00115	1402	0008	BR 00108	
00117	0514		GO	; резерв
00118	0514		GO	
00119	0514		GO	
00120	0614		H	; таблица
00121	0605		E	; сообщений
00122	0704		T	
00123	0200			
00124	0710		З	
00125	0601		А	
00126	0700		П	
00127	0701		Я	
00128	0704		Т	
00129	0615		О	
00130	0610		Й	
00131	0200			
00132	0000			
00133	0614		Н	
00134	0605		Е	
00135	0604		Д	
00136	0615		О	
00137	0700		П	
00138	0705		У	
00139	0703		С	
00140	0704		Т	
00141	0609		И	
00142	0613		М	
00143	0709		Ы	
00144	0610		Й	
00145	0200			

Рис. 25 (продолжение). Внешняя подпрограмма приёма с УСО.

00146	0614	Н	
00147	0615	О	
00148	0613	М	
00149	0605	Е	
00150	0702	Р	
00151	0200		
00152	0705	У	
00153	0703	С	
00154	0615	О	
00155	0200		
00156	0000		
00157	0614	Н	
00158	0605	Е	
00159	0604	Д	
00160	0615	О	
00161	0700	П	
00162	0705	У	
00163	0703	С	
00164	0704	Т	
00165	0609	И	
00166	0613	М	
00167	0709	Ы	
00168	0610	Й	
00169	0200		
00170	0614	Н	
00171	0615	О	
00172	0613	М	
00173	0605	Е	
00174	0702	Р	
00175	0200		
00176	0609	И	
00177	0703	С	
00178	0704	Т	
00179	0615	О	
00180	0714	Ч	
00181	0614	Н	
00182	0609	И	
00183	0611	К	
00184	0601	А	
00185	0200		
00186	0000		; резерв
00187	0000		
00188	0000		
00189	0000		
00190	0000		
00191	0000		
00192	0000		
00193	0000		
00194	0512		

Рис. 25 (окончание). Внешняя подпрограмма приёма с УСО.

Первый блок:

00000	0003	} номер (325)
00001	0205	
00002	0011	} контрольная сумма (2953 ₁₀)
00003	0809	
00004	0000	} количество шагов (232 ₁₀)
00005	1408	
00006	0000	
00007	0512	

Второй блок:

```

00000 1009 0212  BNS #0212,S3 00004
00002 1403 0003  BR 00006
00004 1001 0101  SUB #01,R01
00006 1003 0200  ATOI 0200
00008 1009 0212  BNS #0212,S3 00012
00010 1403 0101  BR 00028
00012 1300 0000  MOV #0000,S00
00014 1301 0607  MOV #0607,S01
00016 1105 1015  MOV R15,R10
00018 0413 0810  ABS R10
00020 1100 0810  ADD R08,R10
00022 1013 0606  JSTT 0606 ; ВЫВОД СИМВ
00024 1013 0304  JSTT 0304 ; "В СТРОКЕ"
00026 0407 0408  JMM 0408
00028 1306 0000  MOV #0000,S06
00030 1307 0015  MOV #0015,S07
00032 0413 1208  MOVH X,R08
00034 0510      BPER 00037
00035 1403 0007  BR 00043
00037 1300 0000  MOV #0000,S00
00039 1301 0704  MOV #0704,S01
00041 1402 0110  BR 00016
00043 1410 1108  BGE R11,R08 00047
00045 1402 0009  BR 00037
00047 1302 0409  MOV #0409,S02
00049 1303 0200  MOV #0200,S03
00051 1100 0809  ADD R08,R09
00053 0413 0209  CMD R09
00055 1003 0200  ATOI 0200
00057 1009 0212  BNS #0212,S3 00061
00059 1403 0003  BR 00063
00061 1402 0302  BR 00012
00063 0413 1208  MOVH X,R0B
00065 0510      BPER 00068
00066 1403 0007  BR 00074
00068 1300 0000  MOV #0000,S00
00070 1301 0812  MOV #0812,S01
00072 1402 0309  BR 00016
00074 1410 1108  BGE R11,R08 00078

```

Рис. 26 (начало). Внешняя подпрограмма вывода на УСО.

00076	1402	0009	BR 00068	
00078	1012	0813	MOV R08,—(R13)	
00080	1013	0712	JSTT 0712	; ВЫЧИСЛ
00082	0413	1208	MOVH X,R08	
00084	0510		BPER 00087	
00085	1403	0007	BR 00093	
00087	1300	0000	MOV #0000,S00	
00089	1301	1010	MOV #1010,S01	
00091	1402	0412	BR 00016	
00093	1013	1114	JSTT 1114	; ОБЪЁМ ОЗУ
00095	1304	0312	MOV #0312,S04	
00097	1305	0708	MOV #0708,S05	
00099	1111	0904	OR S09,S04	
00101	0413	1208	MOVH X,R08	
00103	0904	0810	MOV R08,@R10	
00105	1104	0212	MOV R02,R12	
00107	1015	0813	MOV (R13)+,R08	
00109	1306	1501	MOV #1501,S06	
00111	1307	0100	MOV #0100,S07	
00113	1100	0811	ADD R08,R11	
00115	0413	0211	CMD R11	
00117	0511		RTS	
00118	0514		GO	
00119	0514		GO	
00120	0614		Н	
00121	0605		Е	
00122	0704		Т	
00123	0200			
00124	0710		З	
00125	0601		А	
00126	0700		П	
00127	0701		Я	
00128	0704		Т	
00129	0615		О	
00130	0610		Й	
00131	0200			
00132	0000			
00133	0614		Н	
00134	0605		Е	
00135	0604		Д	
00136	0615		О	
00137	0700		П	
00138	0705		У	
00139	0703		С	
00140	0704		Т	
00141	0609		И	
00142	0613		М	
00143	0709		Ы	
00144	0610		Й	
00145	0200			

Рис.26 (продолжение). Внешняя подпрограмма вывода на УСО.

00146	0614	Н
00147	0615	О
00148	0613	М
00149	0605	Е
00150	0702	Р
0С151	0200	
00152	0705	У
00153	0703	С
00154	0615	О
00155	0200	
00156	0000	
00157	0614	Н
00158	0605	Е
00159	0604	Д
00160	0615	О
00161	0700	П
00162	0705	У
00163	0703	С
00164	0704	Т
00165	0609	И
00166	0613	М
00167	0709	Ы
00168	0610	Й
00169	0200	
00170	0614	Н
00171	0615	О
00172	0613	М
00173	0605	Е
00174	0702	Р
00175	0200	
00176	0700	П
00177	0702	Р
00178	0609	И
00179	0605	Е
00180	0613	М
00181	0614	Н
00182	0609	И
00183	0611	К
00184	0601	А
00185	0200	
00186	0000	
00187	0614	Н
00188	0605	Е
00189	0604	Д
00190	0615	О
00191	0700	П
00192	0705	У
00193	0703	С
00194	0704	Т
00195	0609	И

Рис.26 (продолжение). Внешняя подпрограмма вывода на УСО.

00196	0613	М
00197	0615	О
00198	0605	Е
00199	0200	
00200	0710	З
00201	0614	Н
00202	0601	А
00203	0714	Ч
00204	0605	Е
00205	0614	Н
00206	0609	И
00207	0605	Е
00208	0200	
00209	0707	В
00210	0709	Ы
00211	0702	Р
00212	0601	А
00213	0706	Ж
00214	0605	Е
00215	0614	Н
00216	0609	И
00217	0701	Я
00218	0200	
00219	0000	
00220	0000	
00221	0000	
00222	0000	
00223	0000	
00224	0000	
00225	0000	
00226	0000	
00227	0000	
00228	0000	
00229	0000	
00230	0000	
00231	0512	

Рис. 26 (окончание). Внешняя подпрограмма вывода на УСО.

ние». Подпрограмма выводит на указанный приёмник заданного УСО двоичный эквивалент значения записанного в предложении выражения.

Вариант реализации подпрограммы приведён на [рис. 26](#).

Вывод на приёмник производится командой 1501 01 *«номер приёмника»*, а выбор УСО – так же, как и в подпрограмме приёма.

5.6.5. Следует заметить, что вывод сообщений об ошибках может быть осуществлён с использованием внутренних средств интерпретатора передачей управления входу **ОШИБКА** (JMM 0700). Предварительно в регистр **X** следует занести номер ошибки. Для внешних подпрограмм следует использовать номера ошибок, отсутствующие в [табл. 2](#).

5.7. Организация ввода-вывода текстовой информации

5.7.1. На [рис. 27](#) приведён текст внешней подпрограммы приёма текстовой информации с терминала и записи этого текста в область **DIM** или **COM**, зарезервированную под переменную.

В примере (см. [п. 5.7.3](#)) и тексте внешней подпрограмме присвоен номер 1.

Вызов этой подпрограммы осуществляется предложением:

CALL 1, <переменная>

Переменная может быть одиночной или переменной-массивом. В первом случае для текста резервируется 7 байтов, восьмым будет ограничитель 0000, во втором – столько байтов, сколько резервируется под элементы массива без одного байта.

Контроль длины текста подпрограмма не выполняет.

Первый блок:

00000	0000	} номер подпрограммы (1)
00001	0001	
00002	0002	} контрольная сумма (632 ₁₀)
00003	0708	
00004	0000	} количество шагов (43 ₁₀)
00005	0211	
00006	0000	
00007	0512	

Второй блок:

```

00000 1013 0006  JSTT 0006           ; ПОИСК ПЕР
00002 1012 0913  MOV R09,-(R13)
00004 1012 0113  MOV R01,-(R13)
00006 1012 0813  MOV R08,-(R13)
00008 1013 0010  JSTT 0010           ; ВВОД СТР
00010 1304 0000  MOV #0000,S04       ; запись
00012 0912 0408  MOV S04,@R08       ; ограничителя 0000
00014 1015 0813  MOV (R13)+,R08
00016 1304 0712  MOV #0712,S04       ; начальный адрес
00018 1305 0000  MOV #0000,S05       ; буфера ввода
00020 1000 0208  ADD #02,R08
00022 0913 0310  MOV @R10,S03       ; цикл записи из буфера
00024 0912 0308  MOV S03,@R08       ; в обл. DIM, COM
00026 1000 0108  ADD #01,R08
00028 1000 0110  ADD #01,R10
00030 1009 0000  BNS #0000,S3 00034
00032 1403 0003  BR 00036
00034 1402 0013  BR 00022
00036 1015 0113  MOV (R13)+,R01
00038 1015 0913  MOV (R13)+,R09
00040 0511      RTS
00041 0000
00042 0512      END

```

Рис.27. Внешняя подпрограмма приёма текстовой информации с терминала.

5.7.2. На [рис. 28](#) приведён текст внешней подпрограммы вывода текстовой информации, хранимой в области **DIM** или **COM**, зарезервированной под переменную, на устройство, заданное перед обращением к подпрограмме оператором **PRINT**.

В примере (см. [п. 5.7.3](#)) и тексте внешней подпрограмме присвоен номер 2.

Вызов подпрограммы осуществляется предложением:

CALL 1, <переменная>

5.7.3. На [рис. 29](#) приведён текст программы, иллюстрирующей возможность применения внешних подпрограмм ввода-вывода текстовой информации.

Первый блок:

00000	0000	}	номер подпрограммы (2)
00001	0002		
00002	0001	}	контрольная сумма (299 ₁₀)
00003	0211		
00004	0000	}	количество шагов (19 ₁₀)
00005	0103		
00006	0000		
00007	0512		

Второй блок:

```

00000 1013 0006  JSTT 0006      ; ПОИСК ПЕР
00002 1012 0913  MOV R09,-(R13)
00004 1012 0113  MOV R01,-(R13)
00006 1000 0208  ADD #02,R08
00008 1104 0810  MOV R08,R10
00010 1013 0606  JSTT 0606      ; ВЫВОД СИМВ
00012 1015 0113  MOV (R13)+,R01
00014 1015 0913  MOV (R13)+,R09
00016 0511      RTS
00017 0000
00018 0512      END

```

Рис. 28. Внешняя подпрограмма вывода текстовой информации на устройство вывода.

```

10 DIM A(42,42)
20 PRINT 'ВВЕДИТЕ ПРОИЗВОЛЬНЫЙ ТЕКСТ, НАЖМИТЕ ВК'
30 CALL 1,A(0,0)
40 CALL 2,A(0,0)
50 PRINT: PRINT 'ПРОВЕРЬТЕ ПРАВИЛЬНОСТЬ ВЫВОДА ИНФОРМАЦИИ'
60 END

```

Рис. 29. Пример использования внешних подпрограмм ввода/вывода текстовой информации.

5.8. Использование расширенной памяти объёмом 128 Кбайт

5.8.1. На [рис. 30](#) приведён текст внешней подпрограммы записи значения переменной области **DIM** или **COM** в восьмибайтовую ячейку расширенной памяти.

В тексте и примере (см. [п. 5.8.3](#)) внешней подпрограмме присвоен номер 3.

Вызов подпрограммы осуществляется предложением:

CALL 3, <переменная1>, <переменная2>

где: *переменная1* – переменная области DIM или COM;

переменная2 – номер ячейки расширенной памяти. Нумерация ячеек расширенной памяти сквозная, начиная с 0.

Номер сегмента, которому принадлежит ячейка, вычисляется по формуле:

$$N = \text{int} \frac{G}{1024},$$

где *N* – номер сегмента, *G* – номер ячейки.

Первый блок

00000	0000	} номер подпрограммы (3)
00001	0003	
00002	0003	} контрольная сумма (906 ₁₀)
00003	0810	
00004	0000	} количество шагов (56 ₁₀)
00005	0308	
00006	0000	
00007	0512	

Второй блок

00000	1013	0612	JSTT 0612	; ИМЯ ПЕР
00002	1013	0414	JSTT 0414	; ВЫЗОВ ПЕР
00004	0604		MOV X,Y	
00005	1013	0612	JSTT 0612	; ИМЯ ПЕР
00007	1013	0414	JSTT 0414	; ВЫЗОВ ПЕР
00009	1012	0913	MOV R09, -(R13)	
00011	0413	1209	MOVH X,R09	
00013	1304	0004	MOV #0004,S04	
00015	1305	0000	MOV #0000,S05	
00017	0412	1503	DIVR	; вычисление N
00019	1100	0909	ADD R09,R09	
00021	1100	0909	ADD R09,R09	
00023	1100	0909	ADD R09,R09	
00025	1301	0400	MOV #0400,S01	
00027	1111	0102	OR S01,S02	; вычисление A
00029	1304	1500	MOV #1500,S04	
00031	1111	0504	OR S05,S04	
00033	1305	1312	MOV #1312,S05	
00035	1301	0000	MOV #0000,S01	; запрет внешних
00037	1114	0113	SWA S01,S13	; прерываний
00039	0412	1506	ONSEGM	; подкл. ко 2-й стр.
00041	0412	0009	MOV Y,(R09)	; запись в ячейку
00043	1304	1514	MOV #1514,S04	; первоначальное
00045	1305	1312	MOV #1312,S05	; подключение
00047	0412	1506	ONSEGM	; сегментов
00049	1112	0113	MOV S01,S13	; восстановление маски
00051	1015	0913	MOV (R13)+,R09	; прерываний
00053	0511		RTS	
00054	0000			
00055	0512		END	

Рис. 30. Внешняя подпрограмма записи значения переменной в ячейку расширенной памяти.

Первый блок

00000	0000	} номер подпрограммы (4)
00001	0004	
00002	0003	} контрольная сумма (791 ₁₀)
00003	0107	
00004	0000	} количество шагов (51 ₁₀)
00005	0303	
00006	0000	
00007	0512	

Второй блок

```

00000 1013 0612 JSTT 0612 ; ИМЯ ПЕР
00002 1013 0414 JSTT 0414 ; ВЫЗОВ ПЕР
00004 0413 1209 MOVH X,R09
00006 1304 0004 MOV #0004,S04
00008 1305 0000 MOV #0000,S05
00010 0412 1503 DIVR ; вычисление N
00012 1100 0909 ADD R09,R09
00014 1100 0909 ADD R09,R09
00016 1100 0909 ADD R09,R09
00018 1301 0400 MOV #0400,S01
00020 1111 0102 OR S01,S02 ; вычисление A
00022 1304 1500 MOV #1500,S04
00024 1111 0504 OR S05,S04
00026 1305 1312 MOV #1312,S05
00028 1301 0000 MOV #0000,S01 ; запрет внешних
00030 1114 0113 SWA S01,S13 ; прерываний
00032 0412 1506 ONSEGM ; подкл. ко 2-й стр.
00034 0412 0309 MOV (R09),X
00036 1304 1514 MOV #1514,S04 ; первоначальное
00038 1305 1312 MOV #1312,S05 ; подключение
00040 0412 1506 ONSEGM ; сегментов
00042 1112 0113 MOV S01,S13 ; восстановление маски прер.
00044 1013 0006 JSTT 0006 ; ПОИСК ПЕР
00046 1013 0112 JSTT 0112 ; ЗАП ЗНАЧ
00048 0511 RTS
00049 0000
00050 0512 END

```

Рис. 31. Внешняя подпрограмма записи значения ячейки расширенной памяти в переменную области DIM или COM.

Сегмент с указанной ячейкой памяти подключается ко второй странице, таким образом, абсолютный адрес ячейки в адресном пространстве ДЗ-28 определяется по формуле:

$$A = 16384 + 8 \left(G - 1024 \left(\frac{G}{1024} \right) \right),$$

где A – абсолютный адрес ячейки, C – номер ячейки.

5.8.2. На рис. 31 приведён текст внешней подпрограммы вызова восьмибайтовой ячейки расширенной памяти в переменную области DIM или COM.

В тексте и примере (см. п. 5.8.3) подпрограмме присвоен номер 4.

Вызов подпрограммы осуществляется предложением:

CALL 4, <переменная1>, <переменная2>

где: *переменная1* – номер ячейки расширенной памяти (нумерация сквозная);
переменная2 – переменная области **DIM** или **COM**.

Номер сегмента с указанной ячейкой памяти и абсолютный адрес ячейки в адресном пространстве ДЗ-28 при подключении сегмента ко второй странице вычисляется по формулам, описанным в п.5.8.1.

5.8.3. В качестве примера использования внешних подпрограмм управления расширенной памятью на рис.32 приведён текст программы, осуществляющей проверку перезаписи переменной **A** в переменную **B** области **DIM** через ячейку расширенной памяти. Номер ячейки принимает значения последовательно от 0 до 11263, т.е. проверяются с нулевого по одиннадцатый сегмент расширенной памяти.

```
10 DIM A(0),B(0)
20 LET G=0
30 IF G=11264 THEN PRINT: PRINT 'ПРОВЕРКА ОКОНЧЕНА': STOP
40 LET N=INT(G/1024): LET A=G
50 CALL 3,A,G
60 CALL 4,G,B
70 IF A=B GOTO 90
80 PRINT: PRINT !2*0! 'СЕГМЕНТ'N !5.0! 'ОШИБКА В ЯЧЕЙКЕ'G
90 LET G=G+1: GOTO 30
100 END
```

Рис. 32. Пример использования внешних подпрограмм управления расширенной памятью.

6. ИСПОЛЬЗОВАНИЕ БЭЙСИК-ПРОГРАММ МИКРО-ЭВМ «ЭЛЕКТРОНИКА-60»

6.1. Бэйсик-программы микро-ЭВМ «Электроника-60» (л.2) как правило, могут быть применены при работе с интерпретатором с весьма незначительными изменениями в части управления форматом печати.

6.2. В исключительных случаях применения в программах операторов **DELETE**, **RANDOMIZE** и **OLD**, а также внешней функции **EXF(x)** они должны быть заменены соответствующими операторами:

DELETE – CLEAR P;
OLD – LOAD #1;
EXF(x) – CALL.

Оператор установки начального значения случайного числа **RANDOMIZE** можно заменить циклом, вычисляющим **RND(x)** с прерыванием его в «случайный» момент времени.

6.3. При вводе текста Бэйсик-программы «Электроники-60» с клавиатуры ПМ следует кавычки заменить на апострофы. При загрузке такой программы с перфоленты замену производит интерпретатор.

6.4. Следует иметь в виду, что интерпретатор ДЗ-28, в отличие от интерпретатора «Электроники-60», не переходит в непосредственный режим после выполнения операторов **LIST**, **SAVE**, **LOAD**, **CLEAR** (**DELETE**).

7. СПРАВОЧНОЕ ПРИЛОЖЕНИЕ

7.1. Общие формы записи операторов

7.1.1. При описании операторов в этом разделе используются следующие обозначения:

а) информация, записанная строчными буквами и заключённая в треугольные скобки, задаётся пользователем. Например, в операторе **GOSUB** *<номер строки>* должен быть указан номер строки, скажем, **GOSUB 200**;

б) задание информации, заключённой в квадратные скобки, не обязательно. Например, в операторе загрузки программы с МЛ **LOAD [#0]** параметр **#0** может быть опущен;

в) из параметров, заключённых в фигурные скобки и размещённых один под другим, должен быть выбран один. Например, оператор

$$\text{SKIP} \left\{ \begin{array}{l} \langle \text{выражение} \rangle \\ \langle \text{выражение} \rangle \mathbf{F} \end{array} \right\}$$

может быть записан в двух формах, соответствующих пропуску блоков и файлов;

г) многоточие употребляется для обозначения того, что заключённые в фигурные скобки параметры могут употребляться неоднократно. Например, запись **INPUT** *<переменная>*[{*<переменная>*}...] означает, что оператором **INPUT** можно ввести одну или несколько переменных.

7.1.2. Ниже определены используемые в описаниях термины:

<цифра> ::= 0|1|2|3|4|5|6|7|8|9;

<шестнадцатеричная цифра> ::= 00|01|02|...|15;

<буква> ::= A|B|C|...|X|Y|Z;

<целое> ::= *<цифра>*[{*<цифра>*}...];

<дробная часть> ::= .*<целое>*;

<показатель степени> ::= E { $\left[\begin{array}{l} + \\ - \end{array} \right]$ } *<цифра>* [*<цифра>*].

$$\langle \text{число без знака} \rangle ::= \left\{ \left\{ \begin{array}{l} \langle \text{целое} \rangle \\ \langle \text{дробная часть} \rangle \\ \langle \text{целое} \rangle \langle \text{дробная часть} \rangle \end{array} \right\} \left[\begin{array}{l} \text{показатель} \\ \text{степени} \end{array} \right] \right\};$$

#PI

<переменная> ::= { *<простая переменная>* *<индексированная переменная>* };

<простая переменная> ::= *<буква>* [*<цифра>*];

<индексированная переменная> ::= *<имя массива>* (*<целое>* [, *<целое>*]);

<имя массива> ::= *<буква>* [*<цифра>*];

<массив> ::= *<имя массива>* ();

⟨номер строки⟩ ::= 1|2|...|7999;

⟨функция⟩ ::= { FN ⟨буква⟩
⟨стандартная функция⟩ } (⟨выражение⟩);

⟨стандартная функция⟩ ::= DEG|RAD|SIN|COS|TAN|ASN|ACS|ATN|SGN|SQR|HSN|
HCS|HTN|AHS|AHC|AHC|AHT|ABS|EXP|INT|LGT|LOG|EXT|RND;

⟨цепочка знаков⟩ ::= ⟨любая протяжённость знаков, кроме апострофов, двоеточий
и забоев⟩;

⟨элемент выражения⟩ ::= ⟨число без знака⟩|⟨функция⟩|⟨переменная⟩;

⟨знак арифметической операции⟩ ::= +|-|*|/|-;

⟨выражение⟩ ::= [{ [+]
[-] }] [⟨элемент выражения⟩|⟨выражение⟩ { ⟨знак арифметиче-
ской операции⟩⟨элемент выражения⟩|⟨выражение⟩ } ...];

⟨индекс⟩ ::= ⟨выражение⟩ (значение целой части выражения от 0 до 255 включи-
тельно);

⟨код однобайтовой команды⟩ ::= ⟨шестнадцатеричная цифра⟩⟨шестнадцатерич-
ная цифра⟩;

⟨код двухбайтовой команды⟩ ::= ⟨код однобайтовой команды⟩⟨код однобайтовой
команды⟩;

⟨код команды⟩ ::= ⟨код однобайтовой команды⟩|⟨код двухбайтовой команды⟩.

Примечание. Символ ::= означает «определяется как», символ | – «или».

7.1.3. Общая форма операторов DIM и COM:

DIM { ⟨имя массива⟩ (⟨индекс⟩ [, ⟨индекс⟩]) } [, ...]

COM { ⟨имя массива⟩ (⟨индекс⟩ [, ⟨индекс⟩]) } [, ...]

В качестве индексов записываются максимальные значения индексов элементов определяемого массива. Не допускается запись оператора COM в теле цикла FOR–NEXT.

7.1.4. Общая форма записи оператора определения функции пользователя:

DEF ⟨имя функции⟩ (⟨формальный параметр⟩) = ⟨выражение⟩,

где: ⟨имя функции⟩ ::= FN ⟨буква⟩;

⟨формальный параметр⟩ ::= ⟨простая переменная⟩.

Запись оператора в теле цикла FOR–NEXT не допускается

7.1.5. Оператор REM:

REM ⟨цепочка знаков⟩

Предложение с оператором REM должно быть последним или единственным предложением в строке.

7.1.6. Оператор LET:

[**LET**] ⟨переменная⟩ = ⟨выражение⟩

При отсутствии в исходном тексте слово LET вставляется интерпретатором.

7.1.7. Оператор **DATA**:

DATA [⟨выражение⟩][, ...]

Оператор **DATA** должен быть последним или единственным оператором в строке.

7.1.8. Оператор **READ**:

READ [⟨переменная⟩][, ...]

7.1.9. Оператор **RESTORE**:

RESTORE

7.1.10. Оператор **INPUT**:

INPUT ['⟨цепочка знаков⟩']{⟨переменная⟩}[, ...]

Перед переходом на ввод значения интерпретатор печатает цепочку знаков или, при её отсутствии, вопросительный знак. Ввод должен быть завершён символами **ВК**, **ПС** или ****.

В непосредственном режиме не выполняется.

7.1.11. Оператор **PRINT**:

$$\text{PRINT} \left\{ \begin{array}{l} \left[\begin{array}{l} \left[\begin{array}{l} !\mathbf{F}\langle\text{цифра}\rangle.\langle\text{цифра}\rangle! \\ !\langle\text{цифра}\rangle.\langle\text{цифра}\rangle! \\ !\mathbf{E}! \\ \langle\text{выражение}\rangle \\ '\langle\text{цепочка знаков}\rangle' \\ \mathbf{OPEN} \\ \mathbf{TAB} \langle\text{выражение}\rangle \\ ; \end{array} \right] \end{array} \right] \end{array} \right\} [\dots]$$

Если список оператора заканчивается выражением или апострофом, а также при пустом списке после печати списка выдаётся возврат каретки.

- !E! – машинный формат;
- !F⟨цифра⟩.⟨цифра⟩! – формат с плавающей точкой;
- !⟨цифра⟩.⟨цифра⟩! – формат с фиксированной точкой;
- , – переход к началу следующей зоны;
- ; – пропуск одной позиции.

7.1.12. Оператор **IF**:

$$\mathbf{IF} \left[\begin{array}{l} \langle\text{выражение}\rangle \langle\text{знак отношения}\rangle \langle\text{выражение}\rangle \\ \mathbf{END} [\mathbf{F}] \end{array} \right] \left[\begin{array}{l} \mathbf{THEN} \left[\begin{array}{l} \langle\text{оператор}\rangle \\ \langle\text{номер строки}\rangle \end{array} \right] \\ \mathbf{GOTO} \langle\text{номер строки}\rangle \end{array} \right]$$

где ⟨знак отношения⟩ ::= <|>|=|>|=|=|<|=|=|<|<>|>>

При выполнении условия (или наличии в буфере МЛ записи «КОНЕЦ ФАЙЛА ДАННЫХ» в операторе **IF-END F** или «КОНЕЦ ЛЕНТЫ» в операторе **IF-END**) управление передаётся оператору, записанному после слова **THEN** или строке с указанным номером. В противном случае – строке со следующим номером.

7.1.13. Оператор **FOR**:

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

Первое выражение задаёт начальное значение переменной цикла, второе – граничное, третье – шаг. При отсутствии слова **STEP** шаг полагается равным плюс единице. Значение выражения для шага не должно быть нулевым.

Если условие завершения цикла (см. п. 7.1.14) выполнено сразу при его задании, предложения тела цикла пропускаются.

7.1.14. Оператор **NEXT**:

NEXT *<переменная>*

Оператор ограничивает тело цикла. Цикл выполняется до тех пор, пока при очередном изменении переменной цикла оператором **NEXT** её значение не станет строго больше (при положительном шаге) или строго меньше (при отрицательном шаге) граничного значения, заданного в операторе **FOR** заголовка цикла.

7.1.15. Оператор **GOTO**:

GOTO *<номер строки>*

Управление передаётся строке с заданным номером, или, при её отсутствии, строке с ближайшим бóльшим номером.

7.1.16. Операторы **GOSUB** и **RETURN**:

GOSUB *<номер строки>*

После запоминания адреса возврата выполняется оператор **GOTO** *<номер строки>*. Возврат из подпрограммы производится оператором **RETURN**, общая форма которого: **RETURN**.

7.1.17. Операторы **STOP** и **END**:

STOP

END

Вызывают останов программы. Внутри текста программы рекомендуется использовать оператор **STOP**, а оператор **END** записывать в качестве последнего оператора текста.

7.1.18. Оператор **SAVE**:

SAVE { **#1**
[**#0**][*'<цепочка знаков>'*][*<номер строки>*, *<номер строки>*]
END }

Вариант **SAVE #1** выводит всю программу на перфоленту; **SAVE END** – записывает на МЛ признак (запись) «КОНЕЦ ЛЕНТЫ»; **SAVE [#0]** – текст, ограниченный строками с указанными номерами. При отсутствии номеров строк на МЛ будет записана вся программа. Задание только одного номера – ошибка. Заключённая в апострофы цепочка знаков является именем записываемого программного файла.

7.1.19. Оператор **LOAD**:
$$\text{LOAD} \left\{ \begin{array}{l} \#1 \\ [\#0][\text{'<цепочка знаков>'}][\text{<номер строки>}, \text{<номер строки>}] \end{array} \right\}$$

Вариант **LOAD #1** загружает текст с перфоленты. Загрузка с перфоленты полностью аналогична набору текста с клавиатуры ПМ. **LOAD [#0]** вызывает чтение с МЛ и загрузку программного файла. Заключённая в апострофы цепочка знаков является именем файла. Сравнение имён производится по первым шести символам имени. При отсутствии имени в записи оператора читается очередной программный файл.

Номера́ строк в операторе определяют стираемую перед загрузкой часть программы. При их отсутствии будет стёрта вся программа пользователя.

По завершению выполнения оператора управление передаётся следующей за строкой с **LOAD** строке программы.

7.1.20. Оператор **DATA SAVE**:
$$\text{DATA SAVE} \left\{ \begin{array}{l} \left\{ \begin{array}{l} \text{OPEN} [\text{'<цепочка знаков>'}] \\ \text{END} \end{array} \right\} \\ \left\{ \begin{array}{l} [\#0] \\ \#1 \end{array} \right\} \left\{ \begin{array}{l} \text{<выражение>} \\ \text{<массив>} \end{array} \right\} [, \dots] \end{array} \right\}$$

DATA SAVE #1 – вывод на перфоленту;

DATA SAVE [#0] – запись на МЛ.

DATA SAVE OPEN – записывает на МЛ заголовок файла данных, содержащий (необязательное) имя файла. **DATA SAVE END** формирует запись «КОНЕЦ ФАЙЛА ДАННЫХ». Остальные варианты оператора формируют на МЛ или перфоленте блока данных.

7.1.21. Оператор **DATA LOAD**:
$$\text{DATA LOAD} \left\{ \begin{array}{l} [\#0] \left\{ \begin{array}{l} [\text{'<цепочка знаков>'}] \\ \text{OPEN} \end{array} \right\} \\ \left\{ \begin{array}{l} [\#0] \\ \#1 \end{array} \right\} \left\{ \begin{array}{l} \text{<переменная>} \\ \text{<массив>} \end{array} \right\} [, \dots] \end{array} \right\}$$

Вариант записи оператора **DATA LOAD [#0][<цепочка знаков>']** устанавливает МЛ на начало первого блока данных из файла с заданным именем. По **DATA LOAD OPEN** МЛ остановится после первого заголовка файла данных. Сравнение имён производится по первым шести символам цепочки знаков.

Оператор **DATA LOAD** со списком аргументов читает очередные блоки данных с МЛ ([#0]) или с перфоленты (#1) и присваивает значения указанным переменным и массивам.

7.1.22. Оператор **SKIP**:

SKIP <выражение> [F].

Задаёт пропуск определённого целой частью выражения количества блоков данных или файлов при записи буквы **F**.

Оператор не различает файлы данных и программы. Значение выражения должно быть положительным числом, превышающим единицу.

7.1.23. Оператор **CLEAR**:

$$\text{CLEAR} \left\{ \begin{array}{l} [\text{P}] \langle \text{номер строки} \rangle [, \langle \text{номер строки} \rangle] \\ \text{C} \\ \text{D} \\ \text{F} \\ \text{S} \end{array} \right\}$$

CLEAR C – стирание данных в области **COM** и функций пользователя;

CLEAR D – стирание данных в области **DIM**;

CLEAR F – установка в начальное состояние указателя циклов **FOR – NEXT**;

CLEAR S – установка в начальное состояние указателя стека подпрограмм **GOSUB**;

CLEAR [P] – стирание части программы, определённой указанными номерами.

При отсутствии буквы **P** будет выполнена дополнительная операция перемотки МЛ в начало.

При указании одного нóмера строки будет стёрта только эта строка. Задание оператора **CLEAR [P]** без номеров строк не вызывает никаких действий, за исключением перемотки при отсутствии буквы **P**.

7.1.24. Оператор **REWIND**:

REWIND

Вызывает перемотку МЛ в начало.

7.1.25. Оператор **LIST**:

LIST [*номер строки*] [, *номер строки*]

Оператор выводит на ПМ текст части программы, определённой указанными номерами строк.

При задании одного номера печатается текст только одной строки, при отсутствии номеров – текст всей программы.

7.1.26. Оператор **ON**:

ON *выражение*

Оператор передаёт управление строке с номером, равным целой части значения выражения или, при её отсутствии, строке с ближайшим бóльшим номером.

7.1.27. Оператор **CALL**:

CALL *номер подпрограммы* [*параметры*],

где: *номер подпрограммы* – целое положительное число от 1 до 7999;

параметры – требуемые внешней подпрограммой параметры.

Вызываемая оператором подпрограмма должна быть предварительно загружена в память в процессе начального диалога.

Внешняя подпрограмма должна прочесть все параметры и вернуть по завершению своей работы управление интерпретатору, заслав в регистр **S3** код разделите-

ля, ограничивающего предложение, а в R1 – адрес следующего за разделителем символа.

7.1.28. Оператор **CMD**:

CMD {⟨код команды⟩} [⟨,⟩...]

Вызывает выполнение списка машинных команд.

7.1.29. Оператор **RUN**:

RUN

Запускает программу со строки с наименьшим номером с предварительным стиранием всех данных и установкой указателей в исходное состояние. Оператор вызывает действия, соответствующие выполнению последовательности операторов:

CLEAR C: CLEAR D: CLEAR F: CLEAR S: GOTO 1

Таблица 5

Оператор	Пункты руководства	Номерá типичных ошибок
CALL	7.1.27; 2.15.3	44
CLEAR	7.1.23; 2.14.2	
CMD	7.1.28; 2.15.2	27
COM	7.1.3; 2.3.3	0, 22, 23
DATA	7.1.7; 2.6.2	26
DATA SAVE	7.1.20; 2.12.7; 2.12.8; 2.13.7	50
DATA LOAD	7.1.21; 2.12.9; 2.12.10; 2.13.8	50, 51, 52, 53, 54, 55
DEF	7.1.4; 2.4.2	0, 6, 12, 24
DIM	7.1.3; 2.3.3	0, 22, 23
END	7.1.17; 2.11.1	
FOR	7.1.13; 2.9	30, 31, 33, 34
GOSUB	7.1.16; 2.10; 3.3.5	41, 43
GOTO	7.1.15; 2.8.2; 3.3.5	43
IF	7.1.12; 2.8.3	20, 21
INPUT	7.1.10; 2.6.6	121, 122, 124
LET	7.1.6; 2.6.1	7
LIST	7.1.25; 2.14.3	
LOAD	7.1.19	50, 51, 52, 55
NEXT	7.1.14; 2.9	30, 32
ON	7.1.26; 2.8.5	
PRINT	7.1.11; 2.7	35, 36, 37, 38
READ	7.1.8; 2.6.4	25
REM	7.1.5; 2.1.5	
RESTORE	7.1.9; 2.6.5	
RETURN	7.1.16; 2.10	42
REWIND	7.1.24; 2.12.11	
RUN	7.1.29; 3.3.5	
SAVE	7.1.18; 2.12.4; 2.12.5; 2.13.5	50
SKIP	7.1.22; 2.12.11	50
STOP	7.1.17; 2.11.1	

7.2. Справочные таблицы

7.2.1. В табл. 5 приведён указатель ссылок на пункты руководства, содержащие описание операторов, а также перечень номеров типичных ошибок, которые могут возникнуть при выполнении оператора.

7.2.2. В табл. 6 приведён указатель ссылок для служебных слов словаря интерпретатора.

Таблица 6

Служебное слово	Пункты руководства
OPEN	7.1.11; 7.1.20; 7.1.21; 2.7.8; 2.12.8
STEP	7.1.13; 2.9.2
TAB	7.1.11; 2.7.9
THEN	7.1.12; 2.8.3
TO	7.1.13; 2.9.2
END	7.1.12; 7.1.18; 7.1.20; 2.12.5; 2.12.8; 2.12.12

8. ЛИТЕРАТУРА

1. Ю.Л. Кетков. Программирование на Бэйсике. Москва, «Статистика», 1970 г.
2. Электронная вычислительная машина «Электроника 60» 15ВМ-16. Бэйсик. Программное обеспечение. 0.005.027 ПО2, НИИ «Электроника», 1978 г.
3. Система «ВАНГ-2200В». Руководство для пользователей. Рига, «ЗИНАТНЕ», 1974 г.
4. Ж.П. Ламуатье. Упражнения по программированию на Фортране-IV. «МИР», Москва, 1978 г.

