ГОСУДАРСТВЕННЫЙ СТАНДАРТ
СОЮЗА ССР

ЯЗЫК ПРОГРАММИРОВАНИЯ БЕЙСИК

ГОСТ 27787—88

Издание официальное

КОМИТЕТ СТАНДАРТИЗАЦИИ И МЕТРОЛОГИИ СССР
Москва

ГОСУДАРСТВЕННЫЙ СТАНДАРТ
СОЮЗА ССР

ЯЗЫК ПРОГРАММИРОВАНИЯ БЕЙСИК

ГОСТ 27787—88

Издание официальное

МОСКВА 1992

УДК 681.3.06:006.354 Группа П85

ГОСУДАРСТВЕННЫЙ СТАНДАРТ СОЮЗА ССР

ЯЗЫК ПРОГРАММИРОВАНИЯ БЕЙСИК

Programming language Basic

ГОСТ 27787–88

ОКСТУ 4002

Срок действия с 01.07.89
до 01.07.94

Настоящий стандарт устанавливает:

1) синтаксис программ, написанных на языке Бейсик;

2) форматы и точность данных, а также диапазон представления чисел для данных, поступающих на вход процессора обработки данных, управляемого программой, написанной на языке Бейсик;

3) форматы и точность данных, а также диапазон представления чисел, получаемых в результате выполнения процессором обработки данных программы, написанной на языке Бейсик;

4) семантические правила для интерпретации смысла программ, написанных на языке Бейсик;

5) ошибки и исключительные ситуации, которые должны быть обнаружены, а также способ, при помощи которого эти ошибки и исключительные ситуации должны быть обработаны.

Настоящий стандарт не устанавливает:

1) механизм, при помощи которого программы, написанные на языке Бейсик, преобразуются для использования процессором обработки данных;

2) средства, при помощи которых выполняются программы, написанные на языке Бейсик;

3) состав и форму документации на реализации языке Бейсик и программы, написанные на языке Бейсик.

1. ОСНОВНЫЕ ПОЛОЖЕНИЯ

1.1. Для обеспечения переносимости программ, написанных на языке Бейсик, стандарт строится по концепции "Ядро плюс модули".

Ядро содержит описание операторов и функций, обязательных к реализации на всех ЭВМ, имеющих в составе программного обеспечения язык Бейсик, независимо от их архитектуры и комплектации.

Издание официальное

Каждый модуль содержит описание операторов и функций реализация которых зависит от архитектуры и от комплектации ЭВМ, имеющих в составе программного обеспечения язык Бейсик.

Каждый модуль содержит один или два уровня. Во всех случаях уровни с меньшим номером (нижние) являются собственными подмножествами уровней с большим номером (верхних) внутри того же модуля.

1.2. В стандарте определены следующие способы конструирования допустимых подмножеств языка Бейсик:

1) полный язык Бейсик, состоящий из ядра и высших уровней модулей;

2) подмножества языка Бейсик – любые комбинации ядра и уровней модулей, отличные от полного языка Бейсик;

3) минимальное подмножество должно включать ядро языка Бейсик.

1.3. Каждая реализация должна устанавливать обеспечиваемое ею подмножество. Для выбранного подмножества реализация должна воспринимать все элементы языка.

В дальнейшем, говоря о языке Бейсик, имеется в виду любое допустимое стандартом подмножество языка Бейсик, если явно не оговорено противное.

Каждая реализация языка Бейсик может иметь также дополнительные средства в ядре и модулях. Однако такая реализация будет удовлетворять настоящему стандарту только в том случае, если она обеспечивает синтаксис и семантику операторов и других элементов языка, *описание которых приведено в модулях.*

1.4. Программа удовлетворяет стандарту только, если:

1) каждый оператор, содержащийся в программе, является синтаксически правильным образом, оператора, специфицированного настоящим стандартом;

2) каждый оператор имеет только то значение, которое приведено в настоящем стандарте;

3) *общая совокупность операторов образует пример допустимый* программы, имеющей только то значение, которое приведено в настоящем стандарте.

1.5. Реализация соответствует настоящему стандарту только, если:

1) она допускает и обрабатывает программы, соответствующие настоящему стандарту;

2) она интерпретирует ошибки и исключительные ситуации в соответствии *со спецификациями настоящего стандарта;*

3) ее интерпретация семантики каждого оператора, входящего в состав программы, удовлетворяющей стандарту, соответствует спецификации из настоящего стандарта;

4) ее интерпретация семантики, удовлетворяющей стандарту программы в целом, соответствует спецификации из настоящего стандарта;

5) она допускает ввод, обработку и вывод чисел с точностью не меньшей и в диапазоне не меньшим, чем указано в настоящем стандарте;

6) она сопровождается руководством, в котором однозначно определены действия, предпринимаемые в ответ на обращение к средствам, которые в настоящем стандарте назнваны "неопределенными" или "зависящими от реализации".

1.6. Ряд элементов языка рассчитан на определенные устройства оборудования, функционирование которых должна обеспечить реализация. Реализация оговаривает необходимую минимальную конфигурацию оборудования и специфические компоненты оборудования, функционирование которых она обеспечивает.

Элементы языка, относящиеся к оборудованию, не обеспечиваемому реализацией, могут не включаться в язык, но их отсутствие должно специально оговариваться.

1.7. Реализации, удовлетворяющие настоящему стандарту, могут допускать выполнение программ, написанных на расширенном языке, не требуя вывода сообщений о всех конструкциях, не соответствующих настоящему стандарту. Однако каждый оператор или другой элемент программы, описание которого приведено в реализованном подмножестве настоящего стандарта, не удовлетворяющий описанным здесь синтаксическим правилам, должен вызывать сообщение об ошибке.

Обо всех исключительных ситуациях, описанных в настоящем стандарте, должны выводиться сообщения, если только реализация не содержит механизм, который может быть вызван пользователем для обработки этих ситуаций.

1.8. Некоторые исключительные ситуации (где это указано) могут обрабатываться в соответствии с процедурами, описанными в настоящем стандарте; если таковая процедура отсутствует или не может быть реализована из-за аппаратных ограничений или ограничений, налагаемых операционной средой, то соответствующая исключительная ситуация должна вызывать завершение программы.

Настоящий стандарт не описывает порядок, в котором исключительные ситуации будут обнаруживаться и обрабатываться.

2. СТРУКТУРА ОПИСАНИЯ ЯЗЫКА

Описание каждой конструкции языка построено следующим образом.

1) *Общее описание.* Содержит краткое описание средств Бейсика, которые должны обрабатываться, и указывает общую синтаксическую форму для этих средств.

2) *Синтаксис.* Синтаксис описан при помощи обозначения в виде формы Бэкуса–Наура, детали которой приведены в приложении 1. Несколько слов, составляющих в синтаксисе лексическую единицу, соединены знаком дефис (-).

3) *Примеры.* Краткий список правильных примеров, получаемых при помощи некоторых грамматических правил, представленных в синтаксисе. Нумерация примеров соответствует нумерации грамматических правил и не будет последовательной, если примеры иллюстрируют не все правила.

4) *Семантика.* Семантические правила служат двум целям. Во-первых, они исключают некоторые конструкции, которые допустимы синтаксисом, но по смыслу не удовлетворяют описанию. Во-вторых, они придают смысл упоминаемым конструкциям.

5) *Исключения.* Описываются исключения, возникающие в том случае, когда система, реализующая транслятор с языка Бейсик, обнаруживает, что программа не может быть выполнена или не исполняется в соответствии с настоящим описанием.

6) *Примечания.* Здесь содержатся замечания по некоторым особенностям самого описания и требования по реализации языкового Бейсик-процессора для конкретной операционной поддержки.

В приложении 2 приведены пояснения основных терминов, используемых в настоящем стандарте.

3. ОПИСАНИЕ ЯДРА

3.1. Символы и тексты

3.1.1. *Общее описание*

Набор символов языка Бейсик является подмножеством символов, приведенных в ГОСТ 27465–87. Текст – это последовательность символов, которая используется в программе на языке Бейсик в качестве примечаний (см. п. 3.15), текстовых констант (см. п. 3.3.) или данных (см. п. 3.13.).

3.1.2. *Синтаксис*

1) прописная-буква-латинская = A|B|C|D|E|F|G|H|I|J|K|L|M|N|O|P|Q|R|S|T|U|V|W|X|Y|Z

2) прописная-буква-русская = А|Б|В|Г|Д|Е|Ё|Ж|З|И|Й|К|Л|М|Н|О|П|Р|С|Т|У|Ф|Х|Ц|Ч|Ш|Щ|Ъ|Ы|Ь|Э|Ю|Я

3) цифра = 0|1|2|3|4|5|6|7|8|9

4) символ-текста = кавычки символ-текста-в-кавычках

5) символ-текста-в-кавычках = восклицательный знак|номер| знак-денежной-единицы|процент|коммерческое-И|апостроф|круглая-скобка-левая|круглая-скобка-правая|звездочка|запятая | дробная-черта | обратная-дробная-черта|двоеточие|точка-с-запятой-с-запятой|меньше|равно|больше|вопросительный-знак|подчеркивание|стрелка-вверх|символ-текста-не-в-кавычках (графическое изображение этих символов по ГОСТ 27465–87)

6) символ-текста-не-в-кавычках = пробел|простой-символ-текста

7) простой-символ-текста = знак-плюс|знак-минус|точка|цифра|прописная-буква-латинская|прописная-буква-русская

8) примечания = символ-текста *

9) текст-в-кавычках = кавычки символ-текста-в-кавычках * кавычки

10) текст-не-в-кавычках = простой-символ-текста|простой-символ-текста символ-текст-не-в-кавычках * простой-символ-текста

3.1.3. *Примеры*

8) В ”ПРИМЕЧАНИЯХ” МОЖНО ИСПОЛЬЗОВАТЬ ЛЮБОЙ СИМВОЛ (!, ? = и т.д.)

9) ”В ТЕКСТЕ В КАВЫЧКАХ МОЖНО ИСПОЛЬЗОВАТЬ ЗАПЯТЫЕ, ПРОБЕЛЫ”

10) ALFA+5

3.1.4. *Семантика*

Буквы могут быть прописными буквами латинского или русского алфавита.

Все остальные символы-текста — это любые символы.

3.1.5. *Примечания*

Различные типы символов и текста, описанные в синтаксисе, соответствуют различным видам использования текста в Бейсик-программе. Примечания могут использоваться в операторах примечаний (см. п. 3.15.). Текст-в-кавычках может использоваться в качестве текстовых-констант (см. п. 3.3.). Текст-без-кавычек может использоваться наряду с текстом-в-кавычках в качестве элементов данных (см. п. 3.13); текст-без-кавычек не может начинаться или заканчиваться пробелами.

3.2. Программы

3.2.1. *Общее описание*

Программа на языке Бейсик представляет собой последовательность строк, причем последняя строка программы — это строка, содержащая оператор-конца. Каждая строка программы должна содержать служебное слово. Каждая строка должна иметь уникальный номер-строки, который служит меткой оператора, расположенного в этой строке.

3.2.2. *Синтаксис*

1) программа = блок * последняя-строка

2) блок = строка-с-оператором|цикл-с-шагом

3) строка-с-оператором = номер-строки оператора конец-строки

4) номер-строки = цифра цифра? цифра? цифра?

5) конец-строки — определяется реализацией

6) последняя-строка = номер-строки оператор-конца конец-строки

7) оператор-конца = END

8) оператор = оператор-хранения-данных|оператор-определения-функции | оператор-описания-массива | оператор-вызова-подпрограммы | оператор-безусловного-перехода | оператор-условного перехода | оператор-ввода | оператор-присваивания | оператор-вычисляемого-перехода | оператор-вывода | оператор-запуска-генератора-псевдослучайных-чисел | оператор-рассылки-данных | оператор-примечаний | оператор-восстановления-указателя-рассылки | оператор-возврата-из-подпрограммы | оператор-останова | оператор-объявления-нижней-границы

9) строка = строка-с оператором последняя строка

Синтаксические описания операторов и цикла-с-шагом приведены ниже в соответствующих подразделах.

3.2.3. *Примеры*

6) 999 END

3.2.4. *Семантика*

Программа, написанная на языке Бейсик, состоит из последовательности строк, расположенных в порядке возрастания номеров-строк. Строка с максимальным номером должна быть последней-строкой. Строки программы выполняются поочередно, начиная с первой по порядку, до тех пор, пока:

1) не будет задано некоторое действие, определяемое оператором-управления или блоком-цикла;

2) не произойдет неустранимая ошибка;

3) не выполнится оператор-конца.

Относительно пробелов существует специальное соглашение. За исключением случаев, перечисленных ниже, пробелы могут встречаться в любом месте программы, не влияя на ее выполнение.

Проблемы не допустимы:

1) в начале строки;

2) внутри служебного слова;

3) внутри слова TAB в вызове-табуляции;

4) внутри числовой-константы;

5) внутри номера-строки;

6) внутри функции или внутри идентификатора;

7) внутри операций отношений, состоящих из нескольких символов.

Пробелы в тексте-в-кавычках и в тексте-без-кавычек являются значащими.

До и после каждого служебного слова должен стоять хотя бы один пробел (после служебного слова вместо пробела может стоять конец-строки).

Каждая строка должна начинаться с номера-строки. Целые числа, представляющие номера-строк, должны быть больше нуля; ведущие нули в числе, представляющем номер-строки, игнорируются. Операторы выполняются поочередно в соответствии с возрастанием номеров-строк.

Определение конца-строки зависит от реализации. В качестве конца-строки может использоваться символ возврата каретки или символ возврата каретки, после которого идет символ перевода строки, или конец физической записи.

Строка программы, соответствующей стандарту, может содержать до 72 символов. Признак конца-строки не входит в это число символов.

Оператор-конца не только указывает на физический конец тела программы, но и должен завершать выполнение программы.

3.2.5. *Примечания*

Средства редактирования должны допускать ввод строк программы в любом порядке, а также допускать ввод строк с одинаковыми номерами и строк, содержащих только номер строки. Эти средства должны выполнять сортировку строк программы в надлежащем порядке; в случае дублирования номеров строк должна сохраняться строка, введенная последней. Допускаются реализации, удаляющие строки, содержащие только номер-строки.

3.3. Константы

3.3.1. *Общее описание*

Константы могут быть числовыми и текстовыми.

Числовая константа – это число в десятичной системе счисления. Существуют четыре основных формата представления числовых констант.

1) представление с неявно заданной точкой	sd . . . d
2) представление с явно заданной точкой без задания порядка	sd . . . drd . . . d
3) представление с явно заданной точкой с заданным порядком	sd . . . drd . . . dEsd . . . d
4) представление с неявно заданной точкой с заданным порядком, где d – десятичная цифра; r – знак-точка; s – знак-плюс или знак-минус; E – буква E, показатель порядка.	sd . . . dEsd . . . d

Текстовая константа – это последовательность символов, заключенная в кавычках.

3.3.2. *Синтаксис*

1) числовая-константа = знак? число
2) знак = плюс | минус
3) число = мантисса порядок?
4) мантисса = (целое точка?) | (целое? дробная-часть)
5) *целое = цифра цифра* *
6) дробная-часть = точка целое
7) порядок = E знак? целое
8) текстовая-константа = текст-в-кавычках

3.3.3. *Примеры*

1)
```
–2I
IEI0
5E–I
.4E+I
500
I
.255
```

8) ”XYZ”
”X– 3B2”
”IEI0”

3.3.4. *Семантика*

Значением числовой константы является число, представленное этой константой. Буква Е указывает ”степень числа I0”; при отсутствии знака после Е подразумевается плюс. Пробелы в числовых константах запрещены.

В программе допускаются числовые константы с любым количеством цифр, хотя в конкретных реализациях значения числовых констант округляются до точности, определенной реализацией, но не менее шести значащих десятичных цифр.

Порядок числовой константы также может содержать произвольное число цифр. Если величина ненулевой числовой константы выходит за пределы диапазона, определенного реализацией, фиксируется исключительная ситуация. Минимальный диапазон для числовых констант должен быть в пределах от IE–38 до IE+38. Величины констант меньше машинной точности должны заменяться нулем. Если величина констант больше машинного максимуму, то должно диагностироваться переполнение.

Значением текстовой константы является вся последовательность символов между кавычками, включая пробелы. Длина текстовой константы, т.е. число символов между кавычками, ограничена только длиной строки.

3.3.5. *Исключения*

Вычисление числовой константы вызывает переполнение. Ошибка не является неустранимой, должна существовать восстановительная процедура, заменяющая результат машинным максимумом с определенным знаком, и выдающая соответствующее диагностическое сообщение, после чего вычисления должны продолжаться.

3.3.6. *Примечания*

Возможно переполнение текстовой переменной, если программа пытается присвоить этой переменной текстовую константу длиной, превышающей максимальную длину текста, определяемую реализацией (см. также п. 3.4.4.).

Реализация должна выдавать сообщение о потере точности в константах, значения которых меньше машинного минимума; вычисления после выдачи сообщения должны продолжаться.

3.4. П е р е м е н н ы е

3.4.1. *Общее описание*

Переменные в Бейсике связаны с числовыми или с текстовыми значениями. Числовые переменные могут быть либо простыми переменными, либо ссылками на элементы одно- или двумерного массива, такие ссылки называются индексированными переменными.

Простая-числовая-переменная обозначается буквой, за которой может следовать цифра.

Индексированные числовые-переменные состоят из одной буквы, за которой следует одно или два арифметических выражения, заключенные в круглые скобки.

Для обозначения текстовой-переменной служит буква, за которой следует знак денежной единицы.

Язык Бейсик не требует явных описаний типов переменных. Знак денежной единицы указывает на текстовую переменную, а индекс отличает индексированную переменную от простой-числовой переменной.

3.4.2. *Синтаксис*

1) переменная = числовая-переменная | текстовая-переменная

2) числовая-переменная = простая-числовая-переменная | элемент-числового-массива

3) простая-числовая-переменная = буква цифра?

4) элемент-числового-массива = идентификатор-числового-массива индекс

5) идентификатор-числового-массива = буква

6) индекс = круглая-скобка-левая арифметическое-выражение (запятая арифметическое-выражение)? круглая-скобка-правая

7) текстовая-переменная = буква знак-денежной-единицы

3.4.3. *Примеры*

3) X
   A5

4) V(3)
   W(X,X+Y/2)

7) S¤

3.4.4. *Семантика*

В каждый момент выполнения программы числовой-переменной соответствует единственное числовое значение, а текстовой-переменной – единственное текстовое значение. Значение, соответствующее переменной, может изменяться при выполнении операторов программы. Длина текста, соответствующего текстовой-переменной, может изменяться в процессе выполнения программы от нуля символов (нулевая или пустая текстовая-переменная) до максимального количества символов, допускаемого реализацией. Максимально допустимая длина текстовой-переменной определяется реализацией. При этом реализация должна обеспечить длину не менее 18 символов.

Простые-числовые-переменные и текстовые-переменные объявляются неявно при их первом появлении в программе.

Индексированная переменная ссылается на элемент одномерного или двумерного массива, выбранного по значению индекса (значениям индексов). Значение каждого индекса округляется до ближайщего целого. Индексированные переменные объявляются явно в операторе-объявления-массива. Если индексированные переменные не были явно объявлены в операторе-объявления-массива, то они должны объявляться неявно при их первом появлении в программе. В этом случае при отсутствии

оператора-объявления-нижней-границы индексы принимают значения в диапазоне от нуля до десяти включительно. Выражения, представляющие индексы, должны принимать значения из соответствующего диапазона.

Числовая-переменная, идентификатор которой совпадает (за исключением знака-денежной-единицы) с идентификатором текстовой-переменной, никак с ней не связана.

Начальные значения, присваиваемые переменным, должны определяться реализацией.

3.4.5. *Исключения*

Значения индекса выходит за границы явно или неявно объявленного диапазона (неустранимая ошибка).

3.4.6. *Примечания*

Так как начальные значения, присваиваемые переменным по умолчанию, не специфицируются в настоящем стандарте, а, значит, могут определяться конкретной реализацией, то для того, чтобы программа была переносимой, необходимо каждой переменной в программе явно присвоить значение перед вычислением выражения, в которое входит эта переменная.

Значения переменных, если это не оговорено явно, до выполнения присваивания считаются неопределенными. Тогда при попытке доступа к значению переменной до того, как значение будет явно присвоено, будет зафиксирована искоючительная ситуация.

3.5. В ы р а ж е н и я

3.5.1. *Общее описание*

Выражения могут быть либо арифметическими-выражениями, либо текстовыми-выражениями. Арифметические-выражения образуются из переменных, констант и обращений к функциям при помощи операций сложения, вычитания, умножения, деления и возведения в степень.

Текстовые-выражения составляются либо из текстовых переменных, либо из текстовых констант.

3.5.2. *Синтаксис*

1) выражение = арифметическое-выражение | текстовое-выражение

2) арифметическое-выражение = знак? терм (знак терм)*

3) терм = сомножитель (знак-умножения сомножитель)*

4) сомножитель = первичный (стрелка-вверх первичный)*

5) знак-умножения = звездочка дробная-черта

6) первичный = числовая-переменная | число | обращение-к-числовой-функции | круглая-скобка-левая арифметическое-выражение круглая-скобка-правая

7) обращение-к-числовой-функции = имя-числовой-функции список-аргументов?

8) имя-числовой-функции = функция-определенная-пользователем | встроенная-числовая функция

9) список-аргументов = круглая-скобка-левая аргумент круглая-скобка-правая

10) аргумент = арифметическое-выражение

11) текстовое-выражение = текстовая-переменная | текстовая-константа

3.5.3. Примеры

2) 3*X–Y^2

A(I)+A(2)+A(3)

–X/Y

4) 2^(–X)

6) SQR(X^2+Y^2)

3.5.4. Семантика

Формирование и вычисление арифметических-выражений производится по обычным алгебраическим правилам. Символы стрелка-вверх (Λ), звездочка (*), дробная черта (/), плюс (+) и минус (–) представляют операции возведения в степень, умножения, деления, сложения и вычитания соответственно. Если скобки не меняют порядка вычислений, то операции возведения в степень выполняются первыми, затем выполняются операции умножения и деления и, наконец, операции сложения и вычитания. При отсутствии скобок операции одного старшинства выполняются слева направо.

A–B–C интерпретируются как (A–B)–C;

A^B^C как (A^B)^C;

A/B/C как (A/B)/C и

–A^B как – (A^B).

Если при вычислении арифметического выражения происходит потеря точности, то результат операции, допустившей потерю точности, должен заменяться нулем.

0 |0 по определению равно I.

При вычислении выражения учитываются свойства ассоциативности и коммутативности операций.

При обращении к функции число аргументов должно совпадать с числом параметров – нуль или один), заданных в описании функций.

Обращение-к-функции – это запись вызова ранее определенного алгоритма, в который вместо параметра, заданного в описании функции, подставляется значение аргумента.

Все функции, к которым происходит обращение в выражении, должны быть либо встроены в реализацию, либо описаны при помощи оператора-определения-функции. Результатом вычисления функции, полученным при выполнении ранее описанного алгоритма, является числовое-выражение. Оно замещает обращение-к-функции в выражении.

Значением текстового-выражения является значение текстовой-переменной или текстовой-константы, которая образует текстовое-выражение.

3.5.5. Исключения

Вычисление выражения приводит к делению на нуль. Восстановительная процедура должна заменять результат операции машинным

максимумом со знаком делимого, выдавать диагностическое сообщение и продолжать вычисления.

Вычисление выражения приводит к переполнению. Восстановительная процедура должна заменять результат операции машинным максимумом с алгебраически правильным знаком и выдавать сообщение, после чего вычисления должны продолжаться.

Выполнение операции возведения в степень приводит к возведению отрицательного числа в нецелую степень; выполнение операции возведения в степень приводит к возведению нуля в отрицательную степень. Восстановительная процедура должна заменять результат вычисления положительным машинным максимумом и продолжать вычисления.

3.5.6. *Примечания*

Точность вычисления выражений зависит от реализации, но результат должен иметь не менее 6 значащих цифр.

Метод вычисления степени числа может зависеть от того, является ли показатель степени целым числом или нет. Если показатель степени целое число, то возведение в степень можно заменить многократным умножением, если же показатель степени дробное число, то вычисление степени рекомендуется выполнять при помощи встроенных функций LOG и EXP (см. п. 3.6.).

3.6. Встроенные-функции

3.6.1. *Общее описание*

Ранее описанные алгоритмы для вычисления наиболее широко используемых числовых функций введены в реализацию и поддерживаются реализацией.

3.6.2. *Синтаксис*

Встроенная функция = ABS|ATN|COS|EXP|INT|LOG|RND|SGN|SIN|SQR|TAN

3.6.3. *Семантика*

Значения встроенных-функций и число аргументов каждой функции описаны в табл. 1. Во всех функция X обозначает арифметическое-выражение.

Таблица 1

Функция	Значение функции
ABS (X)	Абсолютное значение X
ATN (X)	Арктангенс X в радианах, т.е. угол, тангенс которого равен X. Диапазоном функции является 0 < ATN (X) < (pi/2), при этом pi является отношением длины окружности круга к его диаметру
COS (X)	Косинус X, где X измеряется в радианах
EXP (X)	Экспонента X, т.е. значение основания натурального логарифма (e = 2.71828. . .), возведенное в степень X; если EXP (X) меньше машинного минимума, то ее значение заменяется нулем
INT (X)	Наибольшее целое, не превышающее X; например, INT (1.3) = 1 и INT (–1.3) = – 2
LOG (X)	Натуральный логарифм X; X должен быть больше нуля

Продолжение табл. 1

Функция	Значение функции
RND	Следующее псевдослучайное число во встроенной-в-реализацию последовательности псевдослучайных чисел, равномерно распределенных в диапазоне $0 < RND < 1$ (см. также п. 3.16)
SGN (X)	Алгебраический "знак" X: -1, если $X < 0$; 0, если $X = 0$ и $+1$, если $X > 0$
SIN (X)	Синус X, где X измеряется в радианах
SQU (X)	Неотрицательное значение квадратного корня их X; X должен быть неотрицательным
TAN (X)	Тангенс X, где X измеряется в радианах

3.6.4. *Исключения*

Значение аргумента функции LOG равно нулю или отрицательное (неустранимая ошибка).

Значение аргумента функции SQR отрицательное (неустранимая ошибка).

Величина функции экспоненты или тангенса больше машинного максимума (значение функции должно заменяться машинным максимумом; вычисления продолжаются).

3.6.5. *Примечания*

Функция RND в отсутствии оператора-запуска-генератора-псевдослучайных-чисел должна генерировать одинаковую последовательность псевдослучайных чисел во время каждого выполнения программы. Это требование выбрано с той целью, чтобы программы, использующие псевдослучайные числа, выполнялись каждый раз с одним результатом.

Если значение функции-экспоненты меньше машинного минимума, то реализация должна выдать сообщение о потере точности, а затем продолжать вычисление.

Алгоритмы вычисления встроенных-функций должны быть построены таким образом, чтобы переполнение, возникшее при промежуточных вычислениях, не вызывало исключительной ситуации, если конечное значение находится в допустимом диапазоне.

3.7. Функции, определенные пользователем

3.7.1. *Общее описание*

Кроме встроенных-функций, Бейсик должен предоставлять пользователю возможность определения новых функций.

Общая синтаксическая форма оператора-определения-функций следующая:

DEF FNx = выражение

или

DEF FNx (параметр) = выражение

где x это одна буква, а параметр – простая-числовая-переменная.

3.7.2. *Синтаксис*

1) оператор-определения-функции = DEF определяемая-числовая-функция списоқ-параметров? знак-равенства арифметическое-выражение

2) определяемая-числовая-функция = FN буква

3) список-параметров = круглая-скобка-левая параметр круглая-скобка-правая

4) параметр = простая-числовая-переменная

3.7.3. *Примеры*

1) DEF FNF (x) = x^4–I

DEF FNA (x) =A*X+B

DEF FNP = 3.I4I59

3.7.4. *Семантика*

Определение функции специфицирует способ вычисления функции в терминах значения выражения, включающего параметр, если он есть в списке параметров и, возможно, другие переменные или константы. Если в определении функции отсутствует список-параметров, то обращения к функции не должны содержать список-аргументов. Если определение функции содержит список-параметров, то обращения к функции должны содержать список-аргументов. В этом случае вычисляется выражение в списке-аргументов, и его значение присваивается параметру из списка параметров в определении функции. Затем вычисляется выражение в определении функции, его значение присваивается функции и является значением функции.

Параметр из списка-параметров, заданный в определении функции, является локальным для этой функции, т.е. переменная с тем же идентификатором, что и внешняя переменная, не имеют ничего общего между собой. Переменные, не описанные как параметры и входящие в арифметическое-выражение, являются одними и теми же как в определении функции, так и вне его.

Определение функции должно находиться в строке с меньшим номером, чем первое обращение к этой функции.

Выражение, заданное в операторе-определения-функции, вычисляется только при обращении к описанной им функции. При выполнении оператора-определения-функции никаких действий в программе не производится и происходят переход на следующую строку.

Определение функции может ссылаться на другие уже определенные функции, но не может ссылаться на текущее определение функции. В программе функции может ссылаться на другие уже определенные функции. В программе функция может быть определена не более одного раза.

3.8. О п е р а т о р-п р и с в а и в а н и я

3.8.1. *Общее описание*

Оператор-присваивания предназначен для присваивания значения выражения переменной. Общая синтаксическая форма следующая:

LET переменная = выражение

3.8.2. *Синтаксис*

1) оператор-присваивания = арифметический-оператор-присваивания текстовый-оператор-присваивания

2) арифметический-оператор-присваивания = LET числовая-переменная знак-равенства арифметическое-выражение

3) текстовый-оператор-присваивания = LET текстовая-переменная знак-равенства текстовое-выражение

3.8.3. *Примеры*

2) LET P = 3.I4I59
   LET A(x, 3) = SIN(Y)*Y+I

3) LET A¤ = "ABC"
   LET A¤ = B¤

3.8.4. *Семантика*

Вычисляется выражение (см. п. 3.5) и его значение присваивается переменной, находящейся слева от знака равенства.

3.8.5. *Исключения*

При присваивании значения текстового выражения текстовой переменной длина результирующего текста превышает допустимую. Возникает переполнение, приводящее к неустранимой ошибке.

3.9. Операторы управления

3.9.1. *Общее описание*

Операторы управления позволяют изменить нормальную последовательность выполнения операторов, указывают номер строки, которая должна выполняться следующей вместо идущей по порядке строки со следующим большим номером-строки.

Оператор-безусловного-перехода.

GO TO номер-строки

допускает безусловную передачу управления.

Оператор-условного-перехода.

IF выр1 отношение выр2 THEN номер-строки

где "выр1" и "выр2" – выражения, "отношение" – это операция отношения. Этот оператор позволяет выполнять условную передачу управления.

Оператор-вызова-подпрограммы и оператор-возврата-из-подпрограммы

GOSUB номер-строки

RETURN

позволяют вызывать подпрограммы.

Оператор-вычисляемого-перехода.

ON выражение GOTO номер строки, . . ., номер-строки позволяет передать управление строке с выбранным номером.

Оператор-останова

STOP

завершает выполнение программы.

3.9.2. *Синтаксис*

Синтаксис определяется следующим образом:

1) оператор-безусловного-перехода = GO пробел* TO номер-строки

2) оператор-условного-перехода = IF выражение-отношения THEN номер-строки

3) выражение отношения = (арифметическое-выражение отношение арифметическое выражение) | (текстовое-выражение отношение-эквивалентности текстовое выражение)

4) отношение = отношение-эквивалентности | меньше | больше | не-меньше | не-больше

5) отношение эквивалентности = знак-равенства | знак-неравенства

6) не-меньше = больше знак-равенства

7) не-больше = меньше знак-равенства

8) не-равно = меньше больше

9) оператор-вызова-подпрограммы = GO пробел* SUB номер-строки

10) оператор-возврата-из-подпрограммы = RETURN

11) оператор-вычисляемого-перехода = ON арифметическое-выражение GO пробел* TO номер-строки (запятая номер-строки)*

12) оператор-останова = STOP

3.9.3. *Примеры*

1) GO TO 999
   GOTO 999

2) IF X > Y+83 THEN 200

9) GO SUB I00
   GOSUB I00

11) 0N L + I0 GO TO 300, 400, 500

12) STOP

3.9.4. *Семантика*

Оператор-безусловного-перехода указывает, что выполнение программы должно быть продолжено со строки, указанной в номере-строки.

Если значение выражения-отношения в операторе-условного-перехода – истина, то выполнение программы будет продолжено со строки с указанным номером-строки; если значение выражения-отношения – ложь, то последовательность выполнения строк не изменяется, т.е. выполняется строка, идущая по порядку за строкой с оператором-условного-перехода.

Отношение не-больше обозначается <=. Аналогично отношение не-меньше обозначается >=. Отношение не-равно обозначается <>.

Два текста считаются эквивалентными тогда и только тогда, когда они имеют одинаковую длину и содержат одинаковые последовательности символов.

Выполнение оператора-вызова-подпрограммы и оператора-возврата-из-подпрограммы может быть описано в терминах стека номеров-строк (но может быть выполнено и по-другому).

Перед выполнением первого оператора-вызова-подпрограммы в программе стек является пустым. Во время выполнения очередного операто-

ра-вызова-подпрограммы номер-строки, в котором расположен оператор-вызова-подпрограммы, помещается в вершину стека, после чего выполнение программы продолжается со строки, объявленной в операторе-вызова-подпрограммы.

При выполнении каждого оператора-возврата-из-подпрограммы из вершины стека извлекается номер строки, после чего программа выполняется со строки, следующей за строкой с данным номером.

Не обязательно, чтобы перед окончанием программы было выполнено одинаковое количество операторов-вызова-подпрограммы и операторов-возврата-из-подпрограммы.

Выражение в операторе-вычисляемого-перехода вычисляется и округляется до ближайшего целого. Это значение затем используется для выбора номера-строки из списка в операторе-вычисляемого-перехода (номера-строк в списке нумеруются слева направо, начиная с 1). Выполнение программы продолжается со строки с выбранным номером-строки.

Все номера-строк в операторах управления должны ссылаться на конкретные строки в программе.

Оператор-останова вызывает окончание программы.

3.9.5. *Исключения*

Попытка выполнения оператора-возврата-из-подпрограммы без выполнения до него соответствующего оператора-вызова-подпрограммы (неустранимая ошибка).

Целое, полученное в выражении в операторе-вычисляемого-перехода, меньше единицы или больше числа элементов в списке номеров-строк (неустранимая ошибка).

3.10. Операторы цикла

3.10.1. *Общее описание*

Оператор-начала-цикла и оператор-конца-цикла служат для создания циклов. Общая синтаксическая форма операторов начала-цикла и конца-цикла следующая:

FOR v = начальное-значение TO конечное-значение STEP шаг NEXT v

где v – простая-числовая-переменная, а начальное-значение, конечное-значение и шаг суть арифметические-выражения; указание STEP шаг является необязательным.

3.10.2. *Синтаксис*

1) цикл-с-шагом = строка-цикла тело-цикла

2) тело-цикла = блок* строка-конца-цикла

3) строка-цикла = номер-строки оператор-начала-цикла конец-строки

4) строка-конца-цикла = номер-строки оператор-конца-цикла конец-строки

5) оператор-начала-цикла = FOR управляющая-переменная равно начальное-значение TO конечное значение (STEP приращение)?

6) управляющая-переменная = простая-числовая-переменная
7) начальное-значение = арифметическое-выражение
8) конечное-значение = арифметическое-выражение
9) шаг = арифметическое-выражение
10) оператор-конца-цикла = NEXT управляющая-переменная

3.10.3. *Примеры*

1) I00 FOR I = ITOI0
   . . . другие блоки или строки
   200 NEXT I
5) FORI = A TO B STEP–I
10) NEXT C7

3.10.4. *Семантика*

Оператор-начала-цикла и оператор-конца-цикла описываются совместно. Физическая последовательность операторов, включающая оператор-начала-цикла и все последующие операторы вплоть до первого оператора-конца-цикла с той же управляющей переменной, называется циклом-с-шагом. Циклы-с-шагом могут быть физически вложенными, т.е. цикл-с-шагом может содержать в себе другой цикл-с-шагом, но они не могут пересекаться, т.е. цикл-с-шагом, который содержит оператор-начала-цикла или оператор-конца-цикла, должен содержать весь цикл-с-шагом, начинающийся или, соответственно, кончающийся этим оператором.

Физически вложенные операторы не могут иметь одну и ту же управляющую-переменную.

При отсутствии указания STEP в операторе-начала-цикла приращение по умолчанию принимается равным +I.

В терминах других операторов можно описать действие оператора-начала-цикла и оператора-конца-цикла следующим образом:

```
FOR = начальное-значение TO конечное-значение STEP шаг (блок)
        NEXT
        эквивалентно
        LET ownI = конечное-значение
        LET own2 = шаг
        LET v    = начальное значение
        . . .
строка I IF (v–ownI)*SGN(own2) > 0 THEN строка 2
        (блок)
        LET v = v+own2
        GOTO строка I
строка 2 REM
```

Здесь v – любая простая-числовая-переменная, ownI и own2 – переменные, связанные с конкретным циклом-с-шагом и не доступные программисту; строкаI и строка 2 – номера-строк, связанные с конкретным циклом-с-шагом и не доступные программисту. Переменные ownI и own2 отличаются от аналогичных переменных, связанных с другими циклами-с-шагом.

Программа может передавать управление внутрь тела-цикла только при помощи оператора-возврата-из-подпрограммы (см. п. 3.9)

3.10.5. *Примечания*

Если в цикле есть приближенные вычисления (например, действия с десятичными дробями в двоичной машине), то цикл будет выполняться в диапазоне граничных значений машинной арифметики. Никаких требований на приближенный результат конечной проверки не накладывается. Нужно заметить, что в большинстве ординарных ситуаций, когда реализованный метод вычислений усекает, а не округляет результат, конструкции вида

FOR x = 0 TO I STEP 0.I

будут выполняться так, как предполагает пользователь, хотя в машине, работающей в двоичной системе счисления, не существует точного представления числа 0.I, а конструкция вида

FORx = I TO 0 STEP 0.I

не будет работать, как ожидается.

Как уже было указано, значение управляющей-переменной при выходе из цикла-с-шагом через оператор-конца-цикла не определено. При выходе из блока-цикла по оператору управления управляющая-переменная должно сохранять значение, котороге она имела при выполнении оператора управления.

Программа, содержащая циклы, будет удовлетворять настоящему стандарту только в том случае, если каждый цикл, которому передано управление, выполняется хотя бы один раз.

3.11. О п е р а т о р-в ы в о д а

3.11.1. *Общее описание*

Оператор-вывода предназначен для вывода данных.

Общая синтаксическая форма оператора-вывода имеет вид:

PRINT элемент р элемент р . . . р элемент

где каждый элемент является либо выражением, либо вызовом-табулятора, либо пробелом, а знак пунктуации (р) является либо запятой, либо точкой с запятой.

3.11.2. *Синтаксис*

1) оператор-вывода = PRINT список-вывода

2) список-вывода = (элемент-вывода разделитель-вывода)* элемент-вывода

3) элемент-вывода = выражение вызов табулятора

4) вызов-табулятора = TAB круглая-скобка-левая арифметическое-выражение круглая-скобка-правая

5) разделитель-вывода = запятая точка-с-запятой

3.11.3. *Примеры*

1) PRINT X
   PRINT X,Y
   PRINT X, Y, Z
   PRINT , , , X

```
PRINT
PRINT "X EQUALS", I0
PRINT X;  (Y+Z)/2
PRINT TAB(I0);  A;  "IS DONE".
```

3.11.4. *Семантика*

Выполнение оператора-вывода создает строку символов для передачи на внешнее устройство. Эта строка символов определяется в результате последовательной обработки каждого элемента-вывода и разделителя-вывода в списке-вывода.

При вычислении арифметических-выражений создается строка символов, содержащая в начале либо пробел, если число положительное, либо знак-минус, если число отрицательное. Далее следует десятичное представление абсолютного значения числа, завершаюшееся пробелом. Положительное десятичное представление числа совпадает с описанием числовых-констант (см. п. 3.3) и используется следующим образом.

В каждой реализации должны быть определены 2 параметра: d – максимальное количество выводимых значащих десятичных цифр числа; e – минимальное количество выводимых цифр порядка числа (см. п. 3.3.1). Значение d должно быть не менее шести, а значение e – не менее двух.

Число, точно представимое целым (d или меньшим) количеством десятичных цифр, выводится в формате представления с неявно заданной точкой без указания порядка.

Все остальные числа выводятся либо в формате представления с явно заданной точкой без задания порядка, либо с явно заданной точкой и с заданным порядком.

Если точность чисел, которые могут быть представлены с d или меньшим количеством цифр без задания порядка, не меньше точности в случае представления этих чисел в формате с заданным порядком, они должны выводиться без задания порядка. Например, если d = 6, то I0^(–6) должно выводиться как 0.00000I.

Числа, представленные в формате с явно заданной точкой без задания порядка, должны выводиться не менее чем с d десятичными значащими цифрами и точкой; последующие нули в дробной части могут быть опущены. В представлении числа, меньшего единицы, должны отсутствовать цифры левее точки. Эта форма требует не менее d+3 символов, включая знак, точку и замыкающий пробел.

Числа, представленные с явно заданной точкой и заданным порядком, должны выводиться в формате:

"значение" E s "порядок"

где s – знак плюс или минус, величина "значения" находится в диапазоне от единицы до десяти и представлена не менее d цифрами точности, а "порядок" содержит от одной до e цифр.

В дробной части "значения" замыкающие нули, могут быть опущены, так же как ведущие нули в "порядке". Точка распечатывается как

часть ”значения”, поэтому выводимое число всего будет содержать d+e+5 символов (знак мантиссы, точка, Е, знак порядка и замыкающий пробел).

При обработке текстовых-выражений генерируется соответствующий текст. При обработке разделителя точка-с-запятой генерируется текст нулевой длины.

Обработка вызова-табуляции или разделителя ”запятая” зависит от ранее сгенерированной строки символов текущего или предыдущего оператора-вывода. ”Текущая строка” (возможно пустая) – это строка символов, сгенерированная после последнего выведенного конца-строки. ”Размер поля” – это число символов, включая символ конца-строки, которые могут быть выведены на одной строке. ”Размер поля” определяется реализацией. ”Указатель позиции” текущей строки – это номер позиции для очередного выводимого в этой строке символа; позиции вывода нумеруются последовательно слева направо, начиная с единицы.

Каждая строка вывода разделена на фиксированное число зон вывода. Число зон и длина каждой зоны определяется реализацией. Все зоны вывода, кроме, возможно, последней в строке, должны иметь одинаковую длину. Эта длина должна быть не менее чем d+e+6 символов. Это нужно, во-первых, для того, чтобы разместить числа, представленные с явно заданной точкой и заданным порядком, и, во-вторых, для перевода механизма вывода с помощью разделителя запятая на следующую зону вывода, как будет описано ниже.

Назначение вызова-табулятора состоит в том, что указатель позиции текущей строки устанавливается на ранее специфицированное значение для вывода следующего элемента-вывода. Для этого аргумент вызова-табуляции вычисляется и округляется до ближайшего целого N. При $N < I$ возникает исключительная ситуация. Если N больше, чем размер поля M, то N преобразуется по формуле

$$N = M*INT((N-I)/M)$$

Если ”указатель позиции” в текущей строке меньше или равен N, то для перемещения указателя позиции в позицию N выводится необходимое количество пробелов. Если указатель позиции больше N, то выводится конец-строки, а в следующей строке выводится N–I пробелов для установления указателя позиции в N.

Обработка разделителя-вывода ”запятая” зависит от ”указателя позиции”. Если данная позиция не находится в последней зоне вывода в строке и не превышает размеры поля, то генерируется один или более пробелов до установления указателя позиции на начало новой зоны вывода в строке. Если ”указатель позиции” находился в последней зоне вывода, то генерируется конец строки. Наконец, если ”указатель позиции” превышает размер поля (как, например, при обработке последнего элемента-вывода, точно заполняющего строку), то генерируется конец-строки, а первая зона вывода в следующей строке заполняется пробелами.

Всякий раз, когда "указатель позиции" больше единицы, а обработка следующего элемента-вывода потребует расширения более чем на одну позицию, перед генерацией символов элемента-вывода будет выведен конец-строки.

Всякий раз, когда при генерации символа во время обработки элемента-вывода "указатель позиции" превысит размер поля более чем на одну позицию, перед этим символом будет выведен конец-строки, "указатель позиции" будет вновь установлен на единицу.

По завершении обработки списка-вывода, если этот список вывода не кончается разделителем-вывода, генерируется последний конец-строки; в противном случае этот последний конец-строки не вырабатывается.

3.11.5. *Исключения*

Вычисление вызова-табулятора генерирует значение меньше единицы (ошибка не является неустранимой, восстановительная процедура должна заменять значение аргумента единицей, после чего выполнение программы должно продолжаться).

3.11.6. *Примечания*

Запятая, используемая как разделитель-вывода, позволяет программисту организовать вывод в табличном формате, который определяется по зонам вывода.

При пустом списке-вывода генерируется конец-строки, формирующий текущую строку вывода. Если в этой строке нет никаких символов, выводится строка из пробелов.

Строка вывода на терминале обычно делится на 5 зон по 15 позиций в каждой.

Строка символов, полученная при выводе значения арифметического-выражения, должна содержать в конце один пробел. В случае, если этот пробел вызывает установку значения указателя позиции, превышающего размер поля, реализация может не выводить этот пробел.

3.12. О п е р а т о р - в в о д а

3.12.1. *Общее описание*

Оператор-ввода позволяет пользователю взаимодействовать с программой в процессе ее выполнения, т.е. присваивать значения переменным программы в режиме диалога или в пакетном режиме. Оператор-ввода дает возможность вводить элементы данных, где элементом данных может быть как числовое, так и текстовое выражение. Общая синтаксическая форма оператора-ввода следующая:

INPUT переменная, . . . , переменная

3.12.2. *Синтаксис*

1) оператор-ввода = INPUT список-переменных
2) список-переменных = переменная (запятая переменная)*
3) подсказка-ввода – определяется реализацией
4) ответ-вводу = список-ввода конец-строки

5) список-ввода = заполненная-заданная-величина (запятая заполненная-заданная-величина)*

6) заполненная-заданная-величина = пробел * заданная-величина пробел *

7) заданная-величина = текст-в-кавычках текст-не-в-кавычках

3.12.3. *Примеры*

1) INPUT x
   INPUT x, A$, Y (2)
5) 2, SMITH, –3
   25,0, –15
7) 3.I4I59

3.12.4. *Семантика*

После того, как при выполнении программы будет получен ответ-вводу, оператор-ввода, присваивает переменным из списка-переменных значения, полученные по порядку из ответа-вводу.

Программа запрашивает у пользователя необходимые данные при помощи подсказки-вводу. Выполнение программы приостанавливается до получения ответа-вводу.

Тип каждой заданной-величины в ответе-вводу должен соответствовать типу переменной, которой присваивается эта величина. Таким образом, тексты-не-в-кавычках, которые являются числовыми константами, должны присваиваться числовым переменным. Строковым переменным должны присваиваться текст-в-кавычках или текст-не-в-кавычках.

Вычисление индексированных-выражений в списке-переменных производится после присваивания значений переменным, предшествующим этим выражениям (т.е. стоящим левее этих выражений) в списке переменных.

Присваиванием значений из ответа-вводу не может производиться до тех пор, пока не будут сделаны следующие проверки:

1) тип каждой заданной-величины должен соответствовать типу переменной;

2) число заданных-величин должно быть равно числу переменных в списке-переменных;

3) каждая из заданных-величин должна находиться в допустимом для нее диапазоне.

3.12.5. *Исключения*

Тип заданной-величины не согласуется с типом переменной, которой она должна быть присвоена (ошибка не является неустранимой, должна существовать восстановительная процедура, разрешающая повторный ответ-вводу).

Число данных в списке-ввода недостаточно (ошибка не является неустранимой, должна существовать восстановительная процедура, разрешающая повторный ответ-вводу).

Число данных превышает число переменных в списке-ввода (ошибка не является неустранимой, должна существовать восстановительная процедура, разрешающая повторный ответ-вводу).

Вычисление арифметической заданной-величины приводит к переполнению (ошибка не является неустранимой, должна существовать восстановительная процедура, разрешающая повторный ответ-вводу).

Присваивание заданной-величины строковой-переменной приводит к переполнению строки (ошибка не является неустранимой, должна существовать восстановительная процедура, разрешающая повторный ответ-вводу).

3.12.6. *Примечания*

Настоящий стандарт требует, чтобы пользователю в диалоговом режиме предоставлялась возможность для повторного ответа-вводу в случае ошибки. Стандарт не требует, чтобы реализация обладала средствами исправления ошибок в ответе-вводу.

Рекомендуется, чтобы подсказка-вводу состояла из знака вопроса и одного пробела.

В случае ошибочного ответа-вводу должно выдаваться сообщение о возникновении исключительной ситуации, и должен быть разрешен повторный ответ-вводу.

Если для присваивания текстовой-переменной вводится текст-не-в-кавычках, то ведущий и завершающий пробелы должны игнорироваться (см. п. 3.1). Если вводится текст-в-кавычках, то пробелы являются значащими (см. п. 3.3).

3.13. Хранение и рассылка данных в программе

3.13.1. *Общее описание*

Оператор-хранения-данных подготавливает последовательность представлений элементов данных для оператора-рассылки-данных. Общая синтаксическая форма оператора-хранения-данных следующая:

DATA заданная-величина, . . . , заданная-величина

где каждая заданная-величина – либо текст-не-в-кавычках (который может быть представлением числовой-константы), либо текст-в-кавычках.

Оператор-рассылки-данных служит для присваивания переменным значений из последовательности данных, созданной оператором-хранения-данных.

Оператор-восстановления-указателя-рассылки позволяет подготовиться к повторному считыванию данных из этой последовательности. Общая синтаксическая форма операторов-рассылки-данных и восстановления-указателя-рассылки следующая:

READ переменная, . . . , переменная

RESTORE

3.13.2. *Синтаксис*

1) оператор-хранения-данных = DATA список-данных

2) список-данных = заданная-величина (запятая заданная-величина)*

3) оператор-рассылки-данных = READ список-переменных

4) оператор-восстановления-указателя-рассылки = RESTORE

3.13.3. *Примеры*

1) DATA 3.14159, PI, 5E–10,

3) READ X, Y, Z
   READ X(I), A(3)

4) RESTORE

3.13.4. *Семантика*

Данные из всей совокупности операторов-хранения-данных собираются в единую последовательность данных. Порядок, в котором данные текстуально появляются в операторах-хранения-данных, определяет их порядок в последовательности данных.

При выполнении оператора-хранения-данных никаких действий не производится, происходит переход на следующую строку.

Оператор-рассылки-данных присваивает значения переменным из списка-переменных в порядке, организованном оператором-хранения-данных. С последовательностью данных связан специальный указатель, который при запуске программы указывает на первую заданную-величину в последовательности данных. При выполнении очередного оператора-рассылки-данных каждой переменной из списка-переменных последовательно присваивается значение заданной-величины, на которую показывает указатель, а указатель при этом передвигается на следующую заданную-величину.

Оператор-восстановления-указателя-рассылки возвращает указатель на начало последовательности, так что при выполнении следующего оператора-рассылки-данных данные будут считываться, начиная с первой заданной-величины из последовательности:

Тип заданной-величины в последовательности данных должен соответствовать типу переменной, которой присваивается эта величина. Числовые переменные требуют присваивания текста-не-в-кавычках, которые являются числовыми-константами, а текстовые-переменные требуют присваивания им текста-в-кавычках или текста-не-в-кавычках. Текст-не-в-кавычках, который является представлением числа, может быть присвоен текстовой-переменной или числовой переменной при помощи оператора-рассылки-данных.

Индексированные выражения из списка-переменных вычисляются после того, как предшествующим переменным (т.е. стоящим левее их) присвоены соответствующие значения.

3.13.5. *Исключения*

Список-переменных в операторе-рассылки-данных требует большее число данных, чем осталось в списке, заданном оператором-хранения-данных (неустранимая ошибка).

Попытка присвоить числовой-переменной текст-в-кавычках или текст-не-в-ковычках, которая не является представлением числовой константы (неустранимая ошибка).

Вычисление числовой-заданной-величины приводит к переполнению Ошибка не является неустранимой, восстановительная процедура должна заменять результат машинным максимумом с соответствующим знаком, вычисления должны продолжаться

Присваивание заданной-величины текстовой-переменной приводит к переполнению текстовой переменной Возникает неустранимая ошибка.

3.13.6. *Примечания*

Ошибки в списке-данных могут генерировать исключительные ситуации во время выполнения операторов рассылки-данных и восстановления-указателя-рассылки.

Реализация должна выдавать сообщение о переполнении как об исключительной ситуации, вычисления должны продолжаться.

Если вычисление числовой-заданной-величины приводит к потере точности, то результат должен заменяться нулем.

3.14. Объявления массивов

3.14.1. *Общее описание*

Оператор-объявления-массива резервирует память под одномерный или двумерный массив. Оператор-объявления-нижней-границы определяет нижнюю границу для всех индексов массива. Использование оператора-объявления-нижней-границы дает возможность объявить в качестве нижней границы индексов массива нуль или единицу. Если нет объявления верхней границы индексов массива, по умолчанию они полагается равной 10. Таким образом, резервируется память для 10 или 11 элементов в одномерном случае и 100 или 121 элемента в двумерном случае в зависимости от установленной нижней границы. Оператор-объявления-массива может объявлять массивы с верхней грани цей индексов, отличной от десяти

Общая синтаксическая форма оператора-объявления-массива следующая.

DIM объявление, . . . , объявление
где каждое объявление имеет вид:
буква (целое)

или

буква (целое, целое)

Общая синтаксическая форма оператора-объявления-нижней-границы следующая:

OPTION BASE n
где n – либо 0, либо I.

3.14.2. *Синтаксис*

1) оператор-объявления-массива = DIM объявление-массива (запятая объявление-массива) *

2) объявление-массива = имя-числового-массива круглая-скобка-левая границы круглая-скобка-правая

3) границы = целое (запятая целое) ?

4) оператор-объявления-нижней-границы = OPTION BASE (0|I)

3.14.3. *Примеры*

1) DIM A (6) , B (I0, I0)

4) OPTION BASE I

3.14.4. *Семантика*

Каждое объявление-массива в операторе-объявления-массива объявляет его размерность. При указании одной границы массив будет одномерным, а при указании двух границ – двумерным.

Кроме того, границы специфицируют максимальные значения для индексных выражений массива.

Объявление массива, если оно присутствует в программе, должно находиться в строке с меньшим номером, чем первое обращение к элементам этого массива. Массивы, не объявленные в операторе-объявления-массива, объявляются неявно одномерными или двумерными, в зависимости от числа индексных выражений. Максимальные значения индексов по умолчанию полагаются равными 10 (см. 3.4).

Оператор-объявления-нижней-границы объявляет минимальное значение для индексов массива; при его отсутствии в программе минимальное значение по умолчанию полагается равным нулю. Если оператор-объявления-нижней-границы в качестве нижней границы значений массива объявляет единицу, то оператор-объявления-массива не может объявлять в качестве верхней границы нуль.

Программа может содержать только один оператор-объявления-нижней-границы.

При выполнении оператора-объявления-массива или оператора-объявления-нижней-границы никаких действий не производится, происходит переход на следующую строку.

Каждый массив в программе можно объявлять только один раз.

3.15. О п е р а т о р - п р и м е ч а н и й

3.15.1. *Общее описание*

Оператор-примечаний позволяет документировать программу.

3.15.2. *Синтаксис*

Оператор-примечаний = REM текст-примечаний

3.15.3. *Примеры*

REM ОКОНЧАТЕЛЬНАЯ ПРОВЕРКА

3.15.4. *Семантика*

При выполнении оператора-примечаний никаких действий не производится, происходит переход к следующей строке.

3.16. З а п у с к - г е н е р а т о р а - п с е в д о с л у ч а й н ы х - ч и с е л

3.16.1. *Общее описание*

Оператор-запуска-генератора-псевдослучайных-чисел вырабатывает определенную реализацией последовательность псевдослучайных чисел, являющихся значениями RND-функции, создавая во время каждого выполнения данной программы различные (и непредсказуемые) последовательности.

3.16.2. *Синтаксис*

Оператор-запуска-генератора псевдослучайных-чисел = RANDOMIZE

3.16.3. *Примеры*

RANDOMIZE

3.16.4. *Семантика*

Выполнение оператора-запуска-генератора-псевдослучайных-чисел вырабатывает новую начальную точку для списка псевдослучайных чисел, используемую функцией RND.

3.16.5. *Примечания*

Если в реализации невозможно обеспечить доступ к устройствам, вырабатывающим случайные данные (например, к таймеру, работающему в реальном масштабе времени), то оператор-запуска-генератора-псевдослучайных-чисел надлежит реализовать с использованием диалога с пользователем.

4. МОДУЛЬ РАСШИРЕНИЯ ОСНОВНЫХ СРЕДСТВ

Этот модуль состоит из двух уровней и содержит описание дополнительных возможностей языка Бейсик, не вошедших в ядро настоящего стандарта. При этом описываются только те возможности, реализация которых не требует включения в состав ЭВМ каких-либо специальных аппаратных средств.

Данный модуль содержит описание только тех средств языка, которые либо не описаны в ядре, либо расширены по сравнению с ядром. При этом пункты, подпункты и перечисления, содержание которых совпадает с описанием, приведенным в ядре, опущены.

4.1. **Уровень 1**

4.1.1. Символы и тексты

4.1.1.1. *Общее описание*

Набор символов дополнительно включает строчные буквы латинского и русского алфавитов.

4.1.1.2. *Синтаксис*

5) символ-текста-в-кавычках = **строчная-буква-латинская** | строчная-буква-русская

11) строчная-буква-латинская = a|b|c|d|e|f|g|h|i|j|k|l|m|o|p|q|r|s|t|u|v|w|x|y|z

12) строчная-буква-русская = а|б|в|г|д|е|ё|ж|з|и|й|к|л|м|н|о|п|р|с|т|у|ф|х|ц|ч| ш|щ|ъ|ы|ь|э|ю|я

4.1.1.3. *Семантика*

Все буквы представляют собой множество прописных и строчных букв латинского и русского алфавитов.

4.1.1.4. *Примечания*

Все строчные буквы латинского и русского алфавитов могут использоваться только в примечаниях и в качестве констант (см. п. 4.1.3.).

4.1.2. Программы

4.1.2.1. *Общее описание*

Расширение основных средств допускает размещение в одной строке нескольких операторов. Кроме того, расширен список допустимых операторов.

4.1.2.2. *Синтаксис*

3) строка-с-оператором = номер-строки пробел * оператор (пробел * двоеточие пробел * оператор) * конец-строки

4) оператор = оператор-определения функции | оператор-определения-типа-переменной | оператор-описания-массива | оператор-условного-перехода | оператор-ввода | оператор-присваивания | оператор-ввода-текста | оператор-вывода-строки-на-печать | оператор-выделения-подстроки | оператор-обработки-ошибок | оператор-вычисляемого-вызова-подпрограммы | оператор-записи-в-оперативную-память | оператор-форматированного-вывода | оператор-выхода-из-подпрограммы-обработки-ошибок | оператор-восстановления-указателя-рассылки | оператор-включения-трассировки | оператор-выключения-трассировки

4.1.2.3. *Примеры*

3) II0 LET A = 5 : PRINTA : STOP

4.1.2.4. *Семантика*

Строки программ выполняются последовательно в соответствии с номерами строк. Операторы, расположенные в одной строке, выполняются по порядку слева направо.

Максимально допустимая длина строк в программах, удовлетворяющих уровню 1, должна быть не менее 255 символов, включая указатель конца-строки.

4.1.2.5. *Примечания*

После некоторых операторов не допускается наличие других операторов, отделенных двоеточием (см. пп. 3.15, 4.1.9).

4.1.3. Константы

4.1.3.1. *Общее описание*

Дополнительно допускаются следующие типы числовых констант:

5) представление с явно заданной sd . . . drd. . . dDsd . . . d точкой с заданным порядком

6) представление с неявно заданной sd . . . dDsd . . . d точкой с заданным порядком

7) представление в виде &Hh . . . h шестнадцатеричного числа

Здесь D – явный символ D, H – явный символ H, а h – десятичная цифра или одна из прописных букв A, B, C, D, E или F.

В конце числовой константы с неявно заданной точкой или с явно заданной точкой без задания порядка может стоять спецификатор типа, указывающий, что данная константа является целой, либо вещественной с одинарной точностью, либо вещественной с удвоенной точностью.

4.1.3.2. *Синтаксис*

4) мантисса = (целое тип?) | (целое точка тип?) | (целое дробная-часть тип?) | шестнадцатеричное-значение

7) порядок = (E знак? целое) | (D знак? целое)

9) тип = процент | восклицательный-знак | номер

10) шестнадцатеричное-значение = &H шестнадцатеричная-цифра шестнадцатеричная-цифра *

11) шестандцатеричная-цифра = цифра |A|B|C|D|E|F

4.1.3.3. *Примеры*

3) 5.7 #
6 %
7!

10) &H2B7

4.1.3.4. *Семантика*

Числовая константа может быть целым числом, либо числом с одинарной точностью, либо числом с удвоенной точностью.

Числовая константа с одинарной точностью – это вещественное число, содержащее в записи не более семи цифр, или
содержащее порядок с буквой E, или
заканчивающееся символом восклицательный знак (!), или
не содержащее указателя типа.

Числовая константа с удвоенной точностью – это вещественное число, содержащее в записи более семи цифр, или
содержащее порядок с буквой D, или
заканчивающееся символом номер (N).

Целая константа – это целое число, заканчивающееся символом процент (%).

Шестнадцатеричные цифры представляются цифрами от 0 до 9, а также буквами, причем A – соответствует 10, B – 11, C – 12, D – 13, E – 14 и F – 15.

Шестнадцатеричные константы должны рассматриваться как константы целого типа.

Диапазон представления чисел зависит от реализации. Диапазон для целых чисел должен быть не менее чем от – 32768 до +32767

Допустимая длина текстовой константы зависит от реализации и должна быть не менее 255 символов.

4.1.3.5. *Примечания*

Тип константы по умолчанию определяется реализацией.

4.1.4. Переменные

4.1.4.1. *Общее описание*

Переменные в Бейсике связаны с числовыми или с текстовыми значениями. Числовые переменные могут быть либо простыми переменными, либо ссылками на элементы массива, такие ссылки называются индексированными переменными.

Идентификаторы простых переменных могут состоять из букв, цифр и начинаются обязательно с буквы.

Идентификаторы индексированных переменных могут состоять из букв, цифр и арифметических выражений, заключенных в круглые

скобки и разделенных запятыми. Идентификаторы индексированных переменных также должны начинаться с буквы.

Идентификаторы переменных могут заканчиваться спецификатором типа.

4.1.4.2. *Синтаксис*

3) простая-числовая-переменная = буква (буква | цифра) * тип?

4) элемент-числового-массива = идентификатор-числового-массива список-индексов

5) идентификатор-числового-массива = буква (буква | цифра) * тип?

6) список-индексов = круглая-скобка-левая арифметическое-выражение (запятая арифметическое-выражение) * круглая-скобка-правая

7) текстовая-переменная = буква (буква | цифра) * знак-денежной-единицы

8) буква = прописная-буква-латинская | прописная-буква-русская

9) элемент-текстового-массива = идентификатор-текстового-массива список-индексов

10) идентификатор-текстового-массива = буква (буква|цифра) * знак-денежной-единицы

4.1.4.3. *Примеры*

3) A2C

MC7 #

4) BI% (I, J+3)

9) D¤(K)

4.1.4.4. *Семантика*

Количество символов, по которым различаются идентификаторы переменных, зависит от реализации, но должно быть не менее 2 .

Длина текстовой переменной может изменяться от 0 (пустая текстовая переменная) до максимальной длины, зависящей от реализации. Максимально допустимая длина текстовой переменной должна быть не менее 255 символов.

Индексированные переменные ссылаются на элементы массивов. Максимальное количество индексов, допускаемое реализацией, должно быть не менее 4 (см. также п. 4.1.15).

Две переменные, имеющие одинаковые идентификаторы, но разные типы, должны рассматриваться как разные переменные.

4.1.5. Выражения

4.1.5.1. *Общее описание*

Выражения могут быть либо арифметическими-выражениями, либо текстовыми-выражениями. Арифметические-выражения образуются из переменных, констант и обращений к функциям при помощи операций сложения, вычитания, умножения, деления, деления нацело, вычисления остатка от деления нацело, возведения в степень, а также логических операций AND, OR, NOT.

Текстовые выражения образуются из текстовых констант, текстовых переменных, текстовых функций и операции конкатенации.

4.1.5.2. *Синтаксис*

5) знак-умножения = звездочка | дробная-черта | обратная-дробная-черта | MOD

10) аргумент = арифметическое-выражение (запятая арифметическое-выражение)*

11) текстовое-выражение = текстовый-элемент (конкатенация текстовый-элемент)*

12) текстовый-элемент = текстовая-переменная | текстовая-функция | элемент-текстового-массива.

13) логическое-выражение = арифметическое-выражение логическая-операция арифметическое-выражение (логическая-операция арифметическое-выражение)*.

14) логическая-операция = AND|OR|NOT

15) конкатенация = знак-плюс

16) текстовая-функция = текстовая-функция-определенная-пользователем|встроенная-текстовая-функция

4.1.5.3. *Примеры*

11) AI¤+ bI¤

12) CHR¤(&H20)
C¤(I, J)

13) (I+5) AND (C–2)

4.1.5.4. *Семантика*

Обратная-дробная-черта обозначает деление нацело. Перед выполнением деления нацело операнды округляются. После выполнения деления дробная часть частного отбрасывается.

MOD обозначает вычисление остатка от деления нацело (деление по модулю). Перед делением операнды приводятся к целому значению. Правила приведения к целому значению определяются реализацией.

Приоритет арифметических операций:

1) возведение в степень;
2) деление, умножение;
3) деление нацело;
4) вычисление остатка от деления нацело;
5) сложение, вычитание.

Все реализации при вычислении арифметических выражений должны обеспечивать следующие правила преобразования типов:

1) если числовая константа одного типа присваивается числовой переменной другого типа, то константа автоматически преобразуется и становится того же типа, что и переменная;

2) при вычислении арифметических выражений все операнды приводятся к одному и тому же типу – типу операнда с максимальной точностью. Если значение результата должно присваиваться переменной, то результат преобразуется к типу переменной;

3) логические операции выполняются только над целыми операндами, поэтому все операнды приводятся к целому типу;

4) если переменной с удвоенной точностью присваивается значение переменной или константы с одинарной точностью, то значащими будут только цифры, соответствующие одинаковой точности.

Логические операции выполняются по правилам, приведенным в *приложении 3*.

Конкатенация текстов – это слияние двух текстов. Результатом конкатенации двух текстов является текст, состоящий из текста, стоящего слева от знака-плюс, дополненного текстом, стоящим справа от знака-плюс.

4.1.5.5. *Исключения*

При конкатенации текстов длина результирующего текста превышает максимальную длину текста, допускаемую реализацией (неустранимая ошибка).

4.1.5.6. *Примечания*

Логические операции выполняются поразрядно над машинными представлениями целых чисел. Эти представления определяются реализацией. Тип результата – целое число.

4.1.6. Встроенные функции

4.1.6.1. *Общее описание*

В расширенной версии языка Бейсик расширен набор числовых функций и, кроме того, дополнительно реализованы текстовые функции. Функции, преобразующие коды в текстовые представления, считаются текстовыми, а преобразующие тексты в коды – числовыми.

4.1.6.2. *Синтаксис*

1) встроенная-функция = встроенная-текстовая-функция| встроенная-числовая-функция

2) встроенная-числовая-функция = CSRLIN|FIX|INSTR|LEN|LPOS|PEEK|POS|RND|VAL

3) встроенная-текстовая-функция = CHR¤ |HEX¤ |INKEY¤ | INPUT¤| LEFT¤|MID¤|RIGHT¤|SPACE¤|STR¤|STRING¤

4.1.6.3. *Семантика*

Значения встроенных-функций и число аргументов каждой функции описаны в табл. 2 и 3. Для всех функций X обозначает произвольное арифметическое выражение, I и J – арифметические выражения, дающие целые результаты, X¤ и Y¤ – текстовые-выражения.

В табл. 2 приведены встроенные-числовые-функции, описание которых отсутствует в п. 3.6.

Таблица 2

Функция	Значение функции
CSRLIN	Возвращает номер строки, в которой находится курсор
FIX(X)	Преобразует X в целое путем отбрасывания дробной части
INSTR (I, X¤, Y¤)	Возвращает номер позиции первого вхождения текста Y¤ в тексте X¤ . I, – необязательный параметр, задающий начало отсчета для поиска

Таблица 2

Функция	Значение функции
LEN(X¤)	Возвращает длину текстовой переменной X¤, т е количество содержащихся в ней символов
LPOS	Возвращает номер позиции (колонки) печатающего узла печатающего устройства
PEEK(I)	Возвращает значение ячейки оперативной памяти с адресом I. Возвращаемое значение – целое число
RND(X)	Если X отсутствует или если X > 0, то функция возвращает очередное сгенерированное случайное число. Если X = 0, то возвращается последнее сгенерированное случайное число. Если X < 0, то генерация случайных чисел запускается заново
VAL(X¤)	Возвращает числовое значение текста X¤. Если первый символ текста X¤ отличен от +, –, или цифры, то возвращается 0

В табл. 3 приведены встроенные-текстовые-функции.

Таблица 3

Функция	Действие
CHR¤(I)	Возвращает символ, код которого соответствует числу I
HEX¤(X)	Возвращает текст, который содержит шестнадцатеричное значение числа X. Число X предварительно округляется
INKEY¤	Возвращает символ, введенный с клавиатуры
INPUT¤(I)	Возвращает текст, содержащий I символов, введенных с клавиатуры. При вводе все управляющие символы, кроме конца-строки, игнорируются
LEFT (X¤, I)	Возвращает текст, содержащий I символов текста X¤, считая с самого левого. I должно быть в диапазоне от 0 до максимально допустимой длины текста. Если I превышает длину текста, то возвращается весь текст; если I = 0, то возвращается пустой текст
MID¤(X¤,I,J)	Возвращает текст, содержащий J символов текста X¤, начиная с 1-го символа, I и J должны быть в диапазоне от 0 до максимально допустимой длины. Аргумент J – необязателен. Если J отсутствует или если в тексте, начиная с позиции I, осталось меньше, чем J символов, то возвращается вся правая часть текста X¤. Если I больше LEN(X¤), то возвращается пустой текст
RIGHT¤(X¤,I)	Возвращает текст, содержащий I символов текста X¤, отсчитывая с конца текста. Если I = 0, то возвращается пустой текст, если I > = LEN(X¤), то возвращается весь текст X¤
SPACE¤(X)	Возвращает текст, содержащий X пробелов. X предварительно округляется
STR¤(X)	Возвращает текстовое представление числа X. X предварительно округляется
STRING¤(I, X)	Возвращает текст, содержащий I одинаковых символов, совпадающих с первым символом текста X¤

4.1.6.5. *Исключения*

Если при преобразовании числа в целое полученное значение выходит из диапазона целых чисел, установленного реализацией, то возникает переполнение. Ошибка не является неустранимой. Реализация должна выводить предупреждающее сообщение и заменять результат на машинный максимум.

Если в текстовых функциях от целого аргумента аргумент превышает максимально допустимое значение кода символа, то возникает ошибка. Ошибка не является неустранимой. Должна производиться замена значения аргумента на максимально допустимое значение кода, выводится диагностическое сообщение и вычисления должны продолжаться. Если аргумент отрицателен, то возникает неустранимая ошибка.

4.1.6.6. *Примечания*

Функция PEEK является аппаратно зависимой, и поэтому ее использование может затруднить переносимость программ. Однако эта функция крайне широко используется в подавляющем большинстве версий языка Бейсик, поэтому она включена в стандарт.

Стандарт не устанавливает ограничения на размер адресуемой ячейки оперативной памяти ЭВМ.

Функция INKEY ¤ возвращает символ, уже находившийся к моменту ее выполнения во внутреннем буфере ввода, определяемого реализацией. Если к моменту выполнения функции в буфере ничего нет, то возвращается пустой текст.

4.1.7. Функции, определенные пользователем

4.1.7.1. *Общее описание*

Общая синтаксическая форма операторов для определения функций следующая:

DEF FNx = выражение

или

DEF FNx (параметры) = выражение,

где x – это последовательность букв и цифр, начинающаяся с буквы, а параметры – это список простых-числовых-переменных, разделенных запятыми.

4.1.7.2. *Синтаксис*

1) оператор-определения-функции = DEF определяемая-текстовая-функция список-параметров?знак-равенства текстовое-выражение

2) определяемая-числовая функция = FN буква*цифра*

3) список-параметров – круглая-скобка-левая параметр (запятая параметр)* круглая-скобка-правая

4) определяемая-текстовая-функция = FN буква*цифра символ-денежной-единицы

4.1.8. Оператор-присваивания

4.1.8.1. *Общее описание*

Оператор-присваивания предназначен для присваивания значения выражения переменной. Общая синтаксическая форма следующая:

LET переменная = выражение или
переменная = выражение

4.1.8.2. *Синтаксис*

2) арифметический-оператор-присваивания = LET? числовая-переменная знак-равенства арифметическое-выражение

3) текстовый-оператор-присваивания = LET? текстовая-переменная знак-равенства текстовое-выражение

4.1.8.3. *Примеры*

2) ALFA = 5*X – 3

3) BI¤= a¤+ CHR (56)

4.1.8.4. *Семантика*

Ключевое слово LET можно опускать. В остальном оператор-присваивания совпадает с описанным в п. 3.8.

4.1.9. Операторы управления

4.1.9.1. *Общее описание*

Оператор-условного-перехода.

IF выр1 отношение выр2 THEN номер-строки или список операторов ELSE номер строки или список операторов.

Здесь "выр1" и "выр2" – выражения, "отношение" – это операция-отношения.

Этот оператор позволяет в зависимости от результатов отношения осуществляться условную передачу управления или выполнять группу операторов. Оператор-условного-перехода должен быть последним оператором в строке (см. п. 4.1.2).

Оператор-вычисляемого-вызова-подпрограммы

ON выражение COSUB номер-строки, номер-строки, ... , номер-строки.

Этот оператор позволяет вызвать подпрограмму с заданным номером.

4.1.9.2. *Синтаксис*

2) оператор-условного-перехода = IF арифметическое-выражение выражение-отношения THEN номер-строки | оператор (пробел * двоеточие пробел * оператор) * (ELSE номер-строки | оператор (пробел * двоеточие пробел * оператор)*))?

3) выражение-отношения = (выражение отношение выражение) (логическая-операция выражение отношение выражение)*

13) оператор-вычисляемого-вызова-подпрограммы = ON арифметическое-выражение GO пробел * SUB номер-строки (запятая номер-строки)*

4.1.9.3. *Примеры*

2) IF A¤< B¤AND X+5 < I0 THEN 500 ELSE STOP
IF X OR Y THEN 200 ELSE 300

4.1.9.4. *Семантика*

Если значение выражения в операторе-условного-перехода – "истина" то выполнение программы будет продолжено с номера-строки,

идущего после слова THEN. Если после ключевого слова THEN стоит последовательность операторов, то после их выполнения будет выполняться строка, идущая после строки с оператором-условного-перехода. Если значение выражения – ”ложь” и оператор содержит ключевое словое ELSE, то управление будет передано строке с указанным номером-строки. Если после слова ELSE стоит последовательность операторов, то после их выполнения будет выполняться строка, идущая после опе-тратора-условного-перехода.

”Ложь” должна иметь в качестве внутреннего представления нуль, а ”истина” должна иметь в качестве внутреннего представления число, отличное от нуля.

При сравнении текстовых-переменных меньшей считается более короткая переменная. Если длина сравниваемых текстовых переменных одинакова, то сравниваются числа, полученные из кодов символов этих переменных. Сравнение ведется посимвольно, начиная с самого левого символа. Меньшее число соответствует меньшей переменной.

Выражение в операторе-вычисляемого-вызова-подпрограммы преобразуется в целое число, которое затем используется для выбора номера-строки из списка в операторе-вычисляемого-вызова-подпрограммы (номера-строк в списке нумеруются слева направо, начиная с I). Потом вызывается подпрограмма с выбранным номером строки.

4.1.9.5. *Исключения*

Целое, полученное при вычислении арифметического-выражения в операторах-вычисляемого-безусловного-перехода и вычисляемого-вызова-под-программы, меньше единицы или больше количества элементов в списке номеров-строк. Ошибка не является неустранимой. Должен выполняться переход к следующей по порядку строке.

4.1.9.6. *Примечания*

Если оператор-условного-перехода используется для проверки на равенство двух арифметических-выражений, дающих вещественный результат, то следует учитывать, что вычисления проводятся с определенной точностью. Поэтому вместо проверки на равенство следует сравнивать абсолютное значение разности этих выражений с машинным минимумом, т.е. вместо записи

IF COS (A) = 0.5 THEN 340

следует использовать запись

IF ABS (COS (A) – 0.5) < 0.00 . . . 0I THEN 340

Оператор-условного-перехода должен быть последним оператором в строке.

4.1.10. О п е р а т о р - ф о р м а т и р о в а н н о г о - в ы в о д а

4.1.10.1. *Общее описание*

Оператор-форматированного-вывода предназначен для генерации форматированного вывода.

Общая синтаксическая форма оператора-форматированного-вывода имеет вид:

PRINT USING стр; элемент р элемент р . . . элемент

где каждый элемент является либо выражением, либо пробелом, р – знак пунктуации (запятая или точка с запятой), а стр – это текстовая-константа или текстовая-переменная, управляющая форматом вывода.

4.1.10.2. *Синтаксис*

Оператор-форматированного-вывода = PRINT *пробел* USING текстовая-константа | текстовая-переменная точка-с-запятой список-печати

4.1.10.3. *Примеры*

PRINT USING„ # #l # # # ”; A, B

4.1.10.4. *Семантика*

Оператор-форматированного-вывода используется для организации форматированного вывода. Управление форматом вывода осуществляется при помощи специальных символов управления форматированием. Символы форматирования для вывода числовых значений приведены в табл. 4.

Таблица 4

Символ	Действие
#	Номер задает обязательно заполняемую позицию каждой цифры. *Если выводимое число имеет меньше цифр, то слева оно дополняется* пробелами
+	Плюс указывает, что перед выводимым числом должен явно стоять его знак: плюс или минус
–	Минус в конце формата указывает, что для выводимых отрицательных чисел знак минус должен быть в конце числа
	Точка отделяет дробную часть выводимого числа от целой части
	Выводимые числа при необходимости округляются до точности определяемой форматом вывода

При выводе текста используются два символа: восклицательный знак (!) указывает, что из данного текста выводится только первый символ; (*n* пробелов) указывает, что из текста выводятся только первые *n* + 2 *символа*.

4.1.10.5. *Исключения*

Выводимое число имеет больше цифр в целой части числа, чем задано в формате вывода. Ошибка не является неустранимой. Должен выполняться полный вывод числа; перед числом вывод символа процент (%)

4.1.11. Оператор-вывода-на-печатающее-устройство

Синтаксис и семантика оператора-вывода-на-печатающее-устройство полностью *совпадают с синтаксисом и семантикой оператора-вывода*, за исключением того, что вместо ключевого слова PRINT используется

ключевое слово LPRINT и вывод осуществляется не на терминах, а на печатающее устройство.

4.1.12. Оператор-ввода

4.1.12.1. *Общее описание*

Оператор-ввода дает возможность вводить элементы данных в режиме диалога, где элементом данных может быть как числовое, так и текстовое выражение. Общая синтаксическая форма оператора-ввода следующая:

INPUT подсказка; переменная, . . . , переменная

где подсказка – это текстовая-константа.

4.1.12.2. *Синтаксис*

1) оператор-ввода = INPUT (подсказка;)? список-ввода

2) подсказка = текстовая-константа

4.1.12.3. *Примеры*

1) INPUT "Введение значения для A и N"; A, N

4.1.12.4. *Семантика*

Подсказка позволяет выводить информацию об ожидаемых вводимых данных.

4.1.13. Оператор-ввода.текста

4.1.13.1. *Общее описание*

Оператор-ввода-текста позволяет вводить данные с терминала, игнорируя все управляющие символы, кроме конца-строки.

Общая синтаксическая форма оператора-ввода-текста следующая:

LINE INPUT (подсказка;)? текстовая-переменная

4.1.13.2. *Синтаксис*

Оператор-ввода-текста = LINE пробел INPUT (подсказка;)? текстовая-переменная

4.1.13.3. *Примеры*

LINE INPUT "Введите ответ"; V¤

4.1.13.4. *Семантика*

Текстовой-переменной присваиваются все вводимые данные до тех пор, пока не будет превышена допустимая длина текста, определяемая реализацией, или пока не будет введен конец-строки.

4.1.14. Объявление массивов

4.1.14.1. *Общее описание*

Оператор-объявления-массивов резервирует память под многомерные массивы.

Общая синтаксическая форма оператора-объявления-массива следующая:

DIM объявление, . . . , объявление

где каждое объявление имеет вид:

имя-переменной (целое, . . . , целое)

4.1.14.2. *Синтаксис*

2) объявление-массива = имя-числового-массива | имя-текстового-массива круглая-скобка-левая-границы круглая-скобка-правая

3) границы = целое (запятая целое)*

4.1.14.3. *Примеры*

1) DIM AI% (10, 3, 2)
   DIM B¤(5, II)

4.1.14.4. *Семантика*

Массивы могут быть текстовыми (элементы массива – текстовые-переменные) и числовыми (элементы массива – числовые-переменные). Числовые массивы могут иметь спецификатор типа, т.е. быть целыми, с одинарной точностью и с удвоенной точностью.

Максимально возможное количество измерений (индексов) должно быть не менее четырех (см. п. 4.1.4.).

4.1.14.5. *Исключения*

Вычисленное значение индекса превышает указанное в операторе-объявления-массивов (неустранимая ошибка).

4.1.15. З а д а н и е т и п о в п е р е м е н н ы х

4.1.15.1. *Общее описание*

Оператор-задания-типов-переменных позволяет задать тип переменных в зависимости от первой буквы имени.

Общая синтаксическая форма оператора следующая:

DEF тип бI–б2, . . . , бk–бl

где тип – это SNG, DBL, STR или INT, а бk–бl – диапазон букв.

4.1.15.2. *Синтаксис*

1) оператор-задания-типов-переменных = DEFтип буква-типа минус буква-типа (запятая буква-типа минус буква-типа)*

2) тип = INT|SGN|DBL|STR

3) буква-типа = прописная-буква-латинская

4.1.15.3. *Примеры*

1) DEFINT A–C, M–P, X– Z

4.1.15.4. *Семантика*

Оператор-задания-типов-переменных указывает, что все переменные, идентификаторы которых начинаются с одной из букв заданного диапазона, являются переменными указанного типа.

При указании диапазона букв буква, стоящая слева от знака минус, в алфавите, должна стоять раньше буквы, стоящей справа от знака минус.

Спецификатор типа имеет более высокий приоритет, чем оператор-задания-типов-переменных.

4.1.16. О п е р а т о р - з а м е н ы - т е к с т а

4.1.16.1. *Общее описание*

Оператор-замены-текста используется для замены части текста на другой текст.

Общая синтаксическая форма оператора следующая:

MID¤(x¤, n, m) = y¤

4.1.16.2. *Синтаксис*

1) оператор-замены-текста = MID¤круглая-скобка-левая текстовая-переменная запятая начальная-позиция (запятая количество)? круглая-скобка-правая знак-равенства текстовое-выражение

2) начальная-позиция = целое

3) количество = целое

4.1.16.3. *Примеры*

1) MID¤(x¤2,5) = y¤

4.1.16.4. *Семантика*

Символы текстовой-переменной, стоящей слева от знака равенства, начиная с *n*-й позиции, заменяются символами из текстовой-переменной, стоящей справа от знака равенства.

Если задано количество, то заменяется указанное количество символов, иначе используется весь текст. Длина текстовой переменной, стоящей слева от знака равенства, не изменяется. Лишние символы текстовой-переменной, стоящей справа от знака равенства, опускаются.

4.1.16.5. *Исключения*

Начальная-позиция или количество, или их сумма превышает длину текста, определенную реализацией (неустранимая ошибка).

4.1.17. О п е р а т о р - з а п и с и - в - о п е р а т и в н у ю - п а м я т ь

4.1.17.1. *Общее описание*

Оператор-записи-в-оперативную-память предназначен для записи целого числа в оперативную память. Общая синтаксическая форма оператора-записи-в-оперативную-память имеет вид:

POKE I, J

4.1.17.2. *Синтаксис*

1) оператор-записи-в-оперативную-память = POKE адрес, данное

2) адрес = целое

3) данное = целое

4.1.17.3. *Примеры*

1) POKE, AI%, B%

4.1.17.4. *Семантика*

Значение данного записывается по указанному адресу.

4.1.17.5. *Исключения*

Значение данного превышает максимальное число, допускаемое реализацией (неустранимая ошибка).

Значение адреса находится вне диапазонов адресов, допускаемых реализацией (неустранимая ошибка)

4.1.17.6. *Примечания*

Оператор-записи-в-оперативную-память является дополнением функции-peek (см. п. 4.1.6). Он также может затруднять переносимость программ.

4.1.18. В с т р о е н н ы е с р е д с т в а о т л а д к и

4.1.18.1. *Общее описание*

Встроенные средства отладки позволяют обрабатывать некоторые ошибочные ситуации, а также выводить на терминал номера выполняемых строк программы.

Переменная-ERR содержит код ошибки.

Переменная-ERL содержит номер-строки, в которой произошла ошибка.

Оператор-обработки-ошибок

ON ERROR GOTOномер-строки

вызывает подпрограмму обработки ошибок.

Оператор-выхода-из-подпрограммы-обработки-ошибок

RESUME аргумент

осуществляет возврат из подпрограммы.

Операторы-включения и выключения-трассировки включают и выключают режим трассировки, т.е. режим вывода на терминал номеров выполняемых программных строк

4.1.18.2. *Синтаксис*

1) оператор-обработки-ошибок = ON пробел ERROR пробел GO пробел* TOномер-строки

2) оператор-выхода-из-подпрограммы-обработки-ошибок = RESUME нуль номер-строки NEXT

3) оператор-включения-трассировки = TRON

4) оператор-выключения-трассировки = TROFF

5) переменная-ERR = ERR

6) переменная-ERL = ERL

4.1.18.3. *Примеры*

1) ON ERROR GOTO I00

2) IF ERL = I20 THEN STOP

4.1.18.4. *Семантика*

Переменная-ERR содержит код ошибки. Список кодов ошибок определяется реализацией.

Переменная-ERL содержит номер строки, в которой произошла ошибка.

Идентификаторы ERR и ERL являются зарезервированными и не могут использоваться в качестве имен-переменных в левой части оператора-присваивания.

Если в программе используется оператор-обработки-ошибок, то при возникновении любой ошибочной ситуации должна вызываться подпрограмма с указанным номером строки.

Возврат из подпрограммы обработки ошибок может осуществляться только при помощи оператора-возврата-из-подпрограммы-обработки-ошибок. При использовании формата RESUME 0 управление снова передается оператору, выполнение которого привело к ошибке.

При использовании формата RESUME NEXT управление передается оператору, выполняемому вслед за ошибочным.

При использовании формата RESUME номер-строки управление передается *строке с заданным номером.*

После выполнения оператора-включения-трассировки на терминал должны поочередно выводиться номера всех выполняемых строк. Вывод строк прекращается после выполнения оператора-выключения трассировки.

4.1.18.5. *Исключения*

Выполнение оператора-возврата-из-подпрограммы-обработки-ошибок без предварительного выполнения оператора-обработки-ошибок (неустранимая ошибка).

4.2. **Уровень 2**

4.2.1. П р о г р а м м ы

4.2.1.1. *Общее описание*

Второй уровень дополнительно содержит цикл-пока, а также операторы, связанные с использованием функциональных клавиш и с позиционированием вывода.

4.2.1.2. *Синтаксис*

2) блок = цикл-пока

4) оператор = оператор-управления-прерыванием-по-клавише | оператор-прерывания-по-клавише

4.2.1.3. *Примеры*

4) 50 KEY ON

4.2.2. К о н с т а н т ы

4.2.2.1. *Общее описание*

В качестве констант дополнительно можно использовать двоичные константы.

4.2.2.2. *Синтаксис*

4) значение = двоичное значение

12) двоичное-значение = &B двоичная-цифра двоичная-цифра*

13) двоичная-цифра = 0|1

4.2.2.4. *Семантика*

Двоичные числа должны рассматриваться как целые числа.

4.2.3. В ы р а ж е н и я

4.2.3.1. *Общее описание*

В выражениях дополнительно можно использовать логические операции XOR, IMP и EQU.

4.2.3.2. *Синтаксис*

14) логическая-операция = XOR|IMP|EQU

4.2.3.3. *Семантика*

Логические операции выполняются по правилам, приведенным в приложении 3.

4.2.4. О п е р а т о р ы у п р а в л е н и я

4.2.4.1. *Общее описание*

К описанным выше операторам управления добавлены операторы-управления прерыванием-по-клавише и прерывания-по-клавише.

Оператор-управления-прерыванием-по-клавише.

KEY (n) ON|OFF|STOP

Этот оператор разрешает или запрещает обращение к подпрограмме при возникновении прерывания от нажатия функциональной клавиши с указанным номером или вызывает останов выполняемой программы.

Оператор-прерывания-по-клавише.

ON KEY GOSUB номер-строки

Этот оператор позволяет вызвать одну из подпрограммы обработки прерываний в зависимости от номера нажатой функциональной клавиши, если ранее прерывание для функциональной клавиши с данным номером было разрешено оператором-управления-прерывания-по клавише.

4.2.4.2. *Синтаксис*

14) оператор-управления-прерыванием-по-клавише = KEY круглая-скобка-левая целое круглая-скобка-правая ON|OFF|STOP

15) оператор-прерывания-по клавише = ON KEY COSUB номер-строки (запятая номер-строки)*

4.2.4.3. *Примеры*

14) KEY (3) ON

15) ON KEY GOSUB 300, 400, 500

4.2.4.4. *Семантика*

Оператор-управления-прерыванием-по-клавише разрешает (параметр ON) или запрещает (параметр OFF) прерывание при нажатии функциональной клавиши на клавиатуре терминала или вызывает останов программы (параметр STOP).

Оператор-прерывания-по-клавише вызывает подпрограмму обработки прерывания с номером-строки, занимающей позицию, соответствующую номеру нажатой функциональной клавиши (см. п. 4.1.9.), если ранее прерывание было разрешено оператором-управления-прерыванием-по-клавише. Если же прерывание было запрещено или была нажата клавиша с номером, превышающим количество номеров-строк в операторе, то программа переходит к выполнению следующего по порядку оператора.

4.2.5. Цикл-пока

4.2.5.1. *Общее описание*

Цикл-пока служит для организации циклов, выполняющихся до тех пор, пока истинно некоторое условие.

Общая синтаксическая форма цикла-пока следующая:

WHILE выражает или отношение

блок

WEND

4.2.5.2. *Синтаксис*

1) цикл-пока = начало-цикла-пока тело-цикла-пока

2) тело-цикла-пока = блок конец-цикла-пока

3) начало-цикла-пока = номер-строки оператор-пока конец-строки

4) оператор-пока = WHILE арифметическое-выражение выражение-отношения

5) конец-цикла-пока = номер-строки WEND

4.2.5.3. *Примеры*

1) I0 WHILE A > 0
   20 A = A – 10
   30 WEND

4.2.5.4. *Семантика*

Циклы-пока могут быть физически вложенными, т.е. цикл-пока может содержать в себе другой цикл-пока. Каждый конец-цикла-пока соответствует ближайшему предыдущему началу-цикла-пока.

Цикл-пока выполняется в том случае, когда значение арифметического-выражения отлично от нуля или если значение выражения-отношения есть истина. Проверка значения арифметического-выражения или выражения отношения должна производиться до выполнения блока, содержащегося в цикле-пока.

5. МОДУЛЬ ГРАФИЧЕСКИХ СРЕДСТВ

Модуль состоит из двух уровней и устанавливает требования для графических средств языка Бейсик. Эти средства предназначены для обеспечения возможности написания программ, которые могут выводить на терминал графическую информацию. Так как графические средства существенно зависят от аппаратуры, то их реализация требует следующих характеристик дисплея:

1) разрешающая способность экрана – не менее 192*256 точек;

2) количество воспроизводимых цветов для цветного дисплея не менее 4.

Размещение вершины координат зависят от реализации.

Ниже в разделе будут использоваться следующие обозначения:

(X, Y) – абсолютные координаты точки. Отсчет ведется относительно вершины координат;

STEP (X, Y) – относительные координаты точки. Отсчет ведется относительно текущего положения графического курсора;

K – номер цвета изображения;

L – номер цвета фона;

M – номер цвета границы (бордюра).

В качестве координат и номеров цветов могут использоваться любые допустимые арифметические-выражения. После выполнения всех вычислений результат автоматически преобразуется в целое число.

5.1. **Уровень 1**

5.1.1. Выбор режима работы

5.1.1.1. *Общее описание*

Режим работы – текстовый, графический или смешанный – устанавливается при помощи оператора-установки-режима. Синтаксическая форма этого оператора следующая:

SCREEN I

где I – номер-режима.

5.1.1.2. *Синтаксис*

1) оператор-установки-режима = SCREEN номер-режима

2) номер-режима = целое

5.1.1.3. *Пример*

1) SCREEN 2

5.1.1.4. *Семантика*

Данный оператор устанавливает режим работы экрана дисплея. При этом реализация, удовлетворяющая стандарту, должна обеспечивать хотя бы один смешанный режим, допускающий одновременный вывод как текстовой, так и графической информации.

Ввод и вывод текстовой информации должны быть допустимы в текстовом и смешанном режимах. Графические операторы выполнимы только в смешанном или в графическом режиме.

5.1.1.5. *Исключения*

Использование оператора в несоответствующем режиме приводит к неустранимой ошибке.

5.1.2. О п е р а т о р - у с т а н о в к и - ц в е т а

5.1.2.1. *Общее описание*

Оператор-установки-цвета устанавливает на экране цвета изображения, фона и границы (бордюра).

Общая синтаксическая форма оператора-установки цвета следующая:

COLOR K, L, M

5.1.2.2. *Синтаксис*

1) оператор-установки-цвета = COLOR список-параметров?

2) список-параметров = номер-цвета? (запятая номер-цвета)? (запятая номер-цвета)?

3) номер-цвета = арифметическое-выражение

5.1.2.3. *Примеры*

1) COLOR, 7

   COLOR I

5.1.2.4. *Семантика*

Оператор-установки-цвета меняет текущие цвета на экране. Если какой-либо из параметров отсутствует, сохраняется текущее значение параметра.

Номера-цветов, порядок задания цветов в операторе и количество воспроизводимых цветов для цветного дисплея определяется реализацией.

5.1.2.5. *Исключения*

Задан несуществующий номер-цвета (ошибка не является неустранимой, состояние цветов определяется реализацией).

5.1.3. Оператор-установки-цвета-точки

5.1.3.1. *Общее описание*

Оператор-установки-цвета-точки устанавливает в графическом режиме цвет точки с заданными координатами.

Общая синтаксическая форма данного оператора следующая:

PSET STEP? (X, Y) (, К)?

5.1.3.2. *Синтаксис*

1) оператор-установки-цвета-точки = PSET список-параметров

2) список-параметров = координаты-точки (запятая номер-цвета-изображения)?

3) координаты-точки = относительные-координаты абсолютные-координаты

4) относительные-координаты = STEP абсолютные-координаты.

5) абсолютные-координаты = круглая-скобка-левая X-координата запятая Y-координата круглая скобка-правая

6) X-координата = арифметическое-выражение

7) Y-координата = арифметическое-выражение

5.1.3.3. *Примеры*

1) PSET (.2*Y^2,X)

5.1.3.4. *Семантика*

Оператор-установки-цвета-точки устанавливает цвет точки с данными координатами.

Координаты могут быть абсолютными (относительно вершины координат) и относительными (относительно текущего положения курсора).

Координатами точки могут быть любые допустимые арифметические-выражения.

При отсутствии номера-изображения сохраняется текущее значение этого параметра.

5.1.3.5. *Исключения*

Неправильно задан номер-цвета-изображения (см. п. 5.1.2).

Координаты точки выходят за границы экрана. Ошибка не является неустранимой. Программа продолжает свою работу.

5.1.4. Изображение отрезков и прямоугольников

5.1.4.1. *Общее описание*

Оператор-изображения-отрезков-и-прямоугольников изображает в графическом режиме отрезки и прямоугольники. Общая синтаксическая форма оператора-изображения-отрезков-и-прямоугольников следующая:

LINE (STEP? (XI, YI))? – STEP? (X2, Y2) ((, К)?, B|BF)?

где STEP? (XI, YI), STEP? (X2, Y2) – координаты концов отрезка (координаты верхней левой и нижней правой точек прямоугольника в случае присутствия параметра B или BF);

B – параметр, объявляющий незаполненный прямоугольник:

BF – параметр, объявляющий прямоугольник, заполненный сплошным цветом.

5.1.4.2. *Синтаксис*

1) оператор-изображения-отрезков-и-прямоугольников = LINE список-параметров

2) список-параметров = координаты-начальной-точки знак-минус координаты-конечной-точки ((запятая номер-цвета-изобаражения)? запятая параметр-прямоугольника)?

3) параметр-прямоугольника = параметр-незаполненного-прямоугольника | параметр заполненного-прямоугольника

4) параметр-незаполненного-прямоугольника = B

5) параметр-заполненного-прямоугольника = BF

6) координаты-начальной-точки = координаты-точки

7) координаты-конечной точки = координаты-точки

5.1.4.3. *Примеры*

1) LINE –STEP (X,Y)
   LINE –STEP (SCALE*3, SCALE*4), BF

5.1.4.4. *Семантика*

Оператор-изображения-отрезков-и-прямоугольников используется для изображения отрезков, а также заполненных и незаполненных прямоугольников.

Если в операторе объявлены координаты двух точек, они задают концы изображаемого отрезка (или верхнюю левую и нижнюю правую точки прямоугольника в случае присутствия в операторе параметров B или BF).

Если в операторе присутствуют координаты только конечной точки, то в качестве начальной точки используется последняя объявленная в графических операторах точка.

Если в операторе отсутствует номер-цвета-изображения, используется текущее значение этого параметра.

При отсутствии параметров B или BF изображается отрезок с заданными координатами концов; иначе, параметр B объявляет незаполненный прямоугольник, а параметр BF – прямоугольник, заполненный сплошным цветом.

5.1.4.5. *Исключения*

Координаты какои-либо из точек выходят за границы экрана. Ошибка не является неустранимой. Изображается видимая часть отрезка или прямоугольника.

5.1.5. Изображение окружностей, друг и эллипсов

5.1.5.1. *Общее описание*

Оператор-изображения-окружности изображает окружность или дуги заданного радиуса и с заданным центром.

Общая синтаксическая форма оператора-изображения-окружности следующая:

CIRCLE STEP? (X, Y), R (, К (начальный-угол-дуги (, конечный-угол-дуги (, эксцентриситет) ?) ?) ?) ?

где R – радиус окружности или дуги; начальный-угол-дуги – точка начала изображения в радианах (если начальный-угол-дуги не задан, то он принимается равным нулю); конечный-угол-дуги – конечная точка изображения в радианах (если конечный-угол-дуги не задан, то он равен 2*PI); углы должны принимать значения из диапазона от –2*PI до +2*PI, где 2*PI = 6.28318; эксцентриситет – величина в диапазоне от 1/260 до 260.

5.1.5.2. *Синтаксис*

1) оператор-изображения-окружности = CIRCLE список-параметров

ров

2) список-параметров = координаты-точки запятая радиус (запятая номер-цвета-изображения (запятая начальный-угол-дуги (запятая конечный-угол-дуги (запятая эксцентриситет) ?) ?) ?) ?

3) радиус = арифметическое-выражение

4) начальный-угол-дуги = арифметическое-выражение

5) конечный-угол-дуги = арифметическое-выражение

6) эксцентриситет = арифметическое-выражение

5.1.5.3. *Примеры*

1) CIRCLE (I00, I00), 75, , -I, –0.0I, 1.2

5.1.5.4. *Семантика*

Оператор-изображения-окружности позволяет изображать на экране окружности, дуги и эллипсы. Координаты-точки задают центр окружности на координатной плоскости. Начальный-угол-дуги и конечный-угол-дуги задают начальную и конечную точку изображения дуги на экране. Если эти параметры имеют отрицательные значения, это означает, что соответствующие конечные точки соединяются отрезками с центром окружности. При отсутствии этих параметров начальной точкой изображения полагается самая правая точка окружности, затем окружность рисуется по часовой стрелке от 0 до 2*PI рад.

Если в операторе объявлен эксцентриситет, то он специфицирует эллипс. Например, если эксцентриситет равен 2, изображается эллипс, высота которого в 2 раза больше чем ширина, полусумма ширины и высоты равна заданному радиусу.

5.1.5.5. *Исключения*

Изображение окружности при заданных параметрах выходит за границы экрана. Ошибка не является неустранимой. Изображается видимая часть окружности.

5.1.6. Окраска области сплошным цветом

5.1.6.1. *Общее описание*

Оператор-окраски-области заполняет область сплошным цветом. Общая синтаксическая форма данного оператора следующая:

PANT STEP? (X, Y) (, К)? (, М)?

где К – номер-цвета-окраски, а М – номер-цвета-изображения границы области.

5.1.6.2. *Синтаксис*

1) оператор-окраски-области = PAINT список-параметров

2) список-параметров = координаты-точки (запятая номер-цвета-окраски) ? (запятая номер-цвета-границы) ?

5.1.6.3. *Примеры*

1) PAINT (I05, 22) , 3, 5

5.1.6.4. *Семантика*

Данный оператор заполняет область заданным цветом. Если граница области должна быть другого цвета, в оператор вводится номер-цвета-границы. В случае, если нет заполнения совпадает с цветом фона, установленного оператором-установки-цвета, можно опустить номер-цвето-заполнения. Координаты-точки задают начальную точку заполнения. В качестве такой точки можно выбрать любую внутреннюю (не граничную) точку области.

5.1.6.5. *Исключения*

Координаты заданной точки находятся вне экрана. Ошибка не является неустранимой. Программа продолжает работу.

5.1.7. Функция-определения-цвета-точки

5.1.7.1. *Общее описание*

Встроенная функция-определения-цвета-точки в качестве результата возвращает номер-цвета-изображения точки с заданными координатами.

5.1.7.2. *Синтаксис*

Функция-определения-цвета-точки = POINT абсолютные-координаты-точки

5.1.7.3. *Примеры*

X = POINT (X0, Y0)

5.1.7.4. *Семантика*

Данная функция возвращает номер-цвета-изображения указанной точки. Если в этой точке ничего не нарисовано, то возвращается номер-цвета-фона.

5.1.8.5. *Исключения*

Координаты точки вне экрана. Ошибка не является неустранимой. Должно выдаваться предупреждающее сообщение, после чего программа продолжает работу.

5.2. **Уровень 2**

5.2.1. Оператор-графических-операций

5.2.1.1. *Общее описание*

Оператор-графических-операций предназначен для построения изображений при помощи специальных операций. Общая синтаксическая форма оператора следующая:

DRAW текстовая-константа

где текстовая-константа содержит список-графических-операций, обозначаемых зарезервированными символами, причем перед некоторыми символами может стоять префикс.

Оператор-графических-операций позволяет в одном тексте задать сразу несколько действий, связанных с выводом графической информации на экран.

5.2.1.2. *Синтаксис*

1) оператор-графических-операций = DRAW список-графических-операций

2) список-графических-операций = (префикс)? зарезервированный-символ параметр ((пробел)?(префикс)? зарезервированный-символ параметр)*

3) префикс = B|N|S

4) зарезервированный-символ = U|D|L|R|E|F|G|H|M

5) параметр = длина | расстояние

6) длина = арифметическое-выражение

7) расстояние = (знак-плюс|знак-минус)? координата-X запятая координата Y

5.2.1.3. *Примеры*

1) DRAW ”UI00 RI20 LI20”
   DRAW ”BEI0”

5.2.1.4. *Семантика*

Список графических-поераций составляет текст длиной до 255 символов. Операции обозначаются зарезервированными символами. Для удобства чтения операции могут разделяться пробелами. Параметр длина задает количество изображаемых точек.

Ниже приведен перечень выполняемых операций:

1) U длина – рисует вертикальную линию вверх от положения графического курсора;

2) D длина – рисует вертикальную линию вниз от графического курсора;

3) L длина – рисует горизонтальную линию влево от графического курсора;

4) R длина – рисует горизонтальную линию вправо от графического курсора;

5) E длина – рисует линию под углом 45°;

6) F длина – рисует линию под углом 315°;

7) G длина – рисует линию под углом 225°;

8) H длина – рисует линию под углом 135°;

9) M (+ или –) координата X, координата Y. Если явно задан знак (+ или –), то координаты отсчитываются от текущего положения графического курсора. В противном случае берутся координаты относительно начала координат.

Префикс B – перемещение графического курсора без рисования следа.

Префикс N – перемещение графического курсора с рисованием следа и возврат графического курсора в исходное положение.

Префикс S – масштаб. Каждая длина умножается на заданный коэффициент. Масштаб может иметь значение от 1 до 255.

Масштабирование действует на все, что рисуется без префикса B.

5.2.1.5. *Исключения*

При выполнении графических-операций изображение выходит за границы экрана. Ошибка не является неустранимой. Программа должна продолжать работу.

5.2.1.6. *Примечания*

Список графических-операций может быть расширен. Допускаются также реализации, позволяющие задавать графические-операций не только текстовыми константами, но и текстовыми переменными.

6. МОДУЛЬ ИНТЕРПРЕТАТОРА

Модуль интерпретатора описывает минимальный набор команд, которые применяет пользователь в диалоге с реализацией языка Бейсик в виде интерпретирующей системы.

Реализация может допускать использование некоторых команд в качестве операторов программы, а некоторых операторов в виде команд. Система должна допускать ввод команд как прописными так и строчными буквами.

Команды, описанные в пп. 6.6–6.8, должны быть реализованы, если в состав ЭВМ входят магнитные диски, а команды, описанные в пп. 6.9–6.11, должны быть реализованы, если в состав ЭВМ входят магнитные ленты.

6.1. К о м а н д а - и н и ц и а л и з а ц и я

6.1.1. *Общее описание*

Команда-инициализация уничтожает программу в памяти.

6.1.2. *Синтаксис*

Команда-инициализации = NEW

6.1.3. *Примеры*

NEW

6.1.4. *Семантика*

Команда-инициализация уничтожает в памяти всю программу, все переменные и закрывает всей файлы. Эту команду необходимо использовать перед тем, как начинать ввод новой программы.

6.2. В ы в о д т е к с т а п р о г р а м м ы

6.2.1. *Общее описание*

По этой команде на экран выводится текст всей программы или ее части. Общая синтаксическая форма команды-вывода-текста следующая:

LIST номер начальной строки – номер конечной строки

6.2.2. *Синтаксис*

1) команда-вывода-текста = LIST (номер-начальной-строки)? (минус)? (номер-конечной-строки)?

2) номер-начальной-строки = точка целое

3) номер-конечной-строки = точка целое

6.2.3. *Примеры*

1) LIST –I75

6.2.4. *Семантика*

Команда-вывода-текста выводит на терминал текст программы, находящейся в данный момент в памяти ЭВМ, в соответствии с указанным диапазоном номеров строк.

Номером-строк может быть любое целое число от 0 до максимального, определяемого реализацией. Если в описании команды не указаны номера-строк, то выводится вся программа. Иначе, если задан единственный номер-строки, то выводится только эта строка. Минус до или после номера-строки означает вывод всего текста до или после строки с указанным номером. Если в команде объявлены два номера-строки, разделенные знаком-минус, то выводятся строки с номерами из указанного диапазона. Если в качестве номера-строки используется точка, она объявляет текущий номер-строки, т.е. последний использованный в команде или введенный номер.

6.2.5. *Исключения*

В команде-вывода-текста объявлены несуществующие строки. Ошибка не является неустранимой. Команда игнорируется.

6.2.6. *Примечания*

Реализация может допускать использование команды-вывода-текста в качестве оператора программы.

6.3. К о м а н д а - з а п у с к а - п р о г р а м м ы

6.3.1. *Общее описание*

Команда-запуска-программы запускает программу на выполнение.

6.3.2. *Синтаксис*

1) команда-запуска-программы = RUN (номер-строки имя-программы)?

2) имя-программы = текстовая-константа | текстовая-переменная

6.3.3. *Примеры*

1) RUN
   RUN I00

6.3.4. *Семантика*

Команда-запуска-программы переводит программу в режим выполнения, начиная выполнение с первой строки программы.

Если задан параметр номер-строки, то выполнение программы начинается со строки с указанным номером.

Если в команде объявлена текстовая-константа или текстовая-переменная, то она интерпретируется как имя файла, содержащего текст программы. Программа загружается в ЭВМ с внешнего устройства (магнитной ленты или магнитного диска) и начинает выполняться.

6.4. Удаление строк программы

6.4.1. *Общее описание*

Команда-удаления-строк-программы удаляет строки текста программы из памяти ЭВМ.

6.4.2. *Синтаксис*

Команда-удаления-строк-программы = DELETE (номер-начальной-строки) ? (минус номер-конечной-строки) ?

6.4.3. *Примеры*

DELETE I00–.

DELETE .

DELETE – 5000

DELETE I35

6.4.4. *Семантика*

Команда-удаления-строк может удалить из памяти часть текста или всю программу. При отсутствии номера-начальной-строки удаляются все строки от начала текста до строки с объявленным номером. При отсутствии номера-конечной-строки удаляются все строки, начиная с указанной. Точка используется для ссылки на текущую строку.

6.4.5. *Исключения*

Указанный в команде номер-строки не существует. Должно выдаваться диагностическое сообщение.

6.4.6. *Примечания*

Допускается сокращенное DEL вместо DELETE.

6.5. Возобновление выполнения программы

6.5.1. *Общее описание*

Команда-возобновления-выполнения-программы вызывает продолжение выполнения программы с того места, где произошел останов программы (после выполнения оператора-останова или после останова с пульта оператора).

6.5.2. *Синтаксис*

Команда-возобновление-выполнения-программы = CONT

6.5.3. *Примеры*

CONT

6.5.4. *Семантика*

Если выполнение программы было прервано встроенным средством прерывания (например, при помощи оператора-останова), то этой командой можно возобновить выполнение программы с той стороны, где она была прервана.

6.5.5. *Примечания*

Если выполнение программы было завершено выполнением оператора-конца (END), то команда-возобновления-выполнения-программы игнорируется.

6.6. Загрузка программы с магнитного диска

6.6.1. *Общее описание*

Команда-загрузки-программы-с-магнитного-диска загружает в оперативную память ЭВМ программу, написанную на языке Бейсик и хранящуюся в файле на магнитном диске.

6.6.2. *Синтаксис*

1) команда-загрузки-программы-с-магнитного-диска = LOAD имя-программы

2) имя-программы = текстовая-константа

6.6.3. *Примеры*

1) LOAD ”MOD2”

6.6.4. *Семантика*

Команда-загрузки-программы-с-диска используется для загрузки программы, написанной на языке Бейсик и сохраненной командой-записи-на-магнитный-диск (см. п. 6.7) на магнитном диске. Имя-программы указывает имя файла, содержащего эту программу.

6.6.5. *Исключения*

Фаила с указанным именем на диске нет. Выдается сообщение ошибке.

6.7. Запись программы на магнитный диск

6.7.1. *Общее описание*

Команда-записи-на-магнитный-диск записывает программу из оперативной памяти ЭВМ в файл на магнитный диск.

6.7.2. *Синтаксис*

Команда-записи-на-магнитный-диск = SAVE имя-программы

6.7.3. *Примеры*

SAVE ”name”

6.7.4. *Семантика*

Команда-записи-на-магнитный-диск записывает текст программы в файл на магнитный диск. Файл получает имя, указанное в имени-программы.

6.7.5. *Исключения*

Диск переполнен. Возникает неустранимая ошибка. Система должна выводить диагностическое сообщение.

6.7.6. *Примечания*

После выполнения команды-записи-на-магнитный-диск программа должна сохраняться в оперативной памяти ЭВМ.

Длина имени-программы определяется реализацией.

6.8. Вывод каталога диска

6.8.1. *Общее описание*

Эта команда выводит список имен файлов, хранящихся на магнитном диске.

6.8.2. *Синтаксис*

Вывод-каталога-диска = FILES

6.8.3. *Примеры*

FILES

6.8.4. *Семантика*

Команда-вывода-каталога-диска выводит на терминал имена всех файлов, которые существуют на диске.

6.8.5. *Примечания*

Формат вывода каталога определяется реализацией.

6.9. Управление накопителем на магнигной ленте

6.9.1. *Общее описание*

Эта команда включает и выключает мотор накопителя на магнитной ленте.

6.9.2. *Синтаксис*

Команда-управления-магнитофоном = MOTOR ON | OFF

6.9.3. *Примеры*

MOTOR ON

6.9.4. *Семантика*

Эта команда включает или выключает мотор магнитофона.

6.10. Загрузка программы с магнитной ленты

6.10.1. *Общее описание*

Эта команда производит загрузку программы на языке Бейсик с магнитной ленты в оперативную память ЭВМ.

6.10.2. *Синтаксис*

Команда-загрузки-программы-с-магнитной-ленты = CLOAD имя-программы?

6.10.3. *Примеры*

CLOAD ".2FJLE"

6.10.4. *Семантика*

Команда-загрузки-программы-с-магнитной-ленты загружает программу с указанным именем с магнитной ленты.

6.10.5. *Исключения*

Программа с указанным именем отсутствует на магнитной ленте. Система должна выводить диагностическое сообщение.

6.10.6. *Примечания*

Длина имени файла определяется реализацией.

6.11. Запись программы на магнитную ленту

6.11.1. *Общее описание*

Эта команда записывает текст программы, написанной на языке Бейсик и хранящейся в оперативной памяти ЭВМ, на магнитную ленту.

6.11.2. *Синтаксис*

Команда-записи-на-магнитную-ленту =CSAVE имя-программы

6.11.3. *Примеры*

CSAVE "I00I"

6.11.4. *Семантика*

Производится запись текста программы на магнитную ленту в файл с заданным именем.

6.11.5. *Исключения*

На магнитной ленте нет места. Ошибка не является неустранимой.

Программа должна сохраняться в памяти ЭВМ. Система должна выводить диагностическое сообщение.

7. МОДУЛЬ РАБОТЫ С МАГНИТНЫМИ ЛЕНТАМИ

Данный модуль содержит один уровень и устанавливает синтаксис и семантику операторов для работы с информацией, хранящейся на магнитной ленте.

7.1. Оператор-открытия-файла

7.1.1. *Общее описание*

Оператор-открытия-файла подготавливает файл для операций ввода/вывода и устанавливает режим чтения или записи

7.1.2. *Синтаксис*

1) оператор-открытия-файла = OPEN пробел кавычки режим кавычки запятая номер номер-файла запятая имя-файла

2) режим = I|O

3) номер-файла = целое

4) имя-файла = текстовая-константа | текстовая-переменная

7.1.3. *Примеры*

1) OPEN ”O”, # 2, ”CASS:SSFILE”.

7.1.4. *Семантика*

Номер-файла – это число, связанное с файлом все время, пока файл находится открытым, и используемое другими операторами ввода-вывода при обращении к файлу.

Имя-файла должно начинаться с символов CAS:

Оператор-открытия-файла должен быть выполнен до первого оператора ввода или вывода с использованием данного файла.

Если имя-файла отсутствует, то открывается следующий по порядку файл на магнитной ленте.

7.1.5. *Исключения*

Файл, открываемый для чтения, не найден (неустранимая ошибка).

7.1.6. *Примечания*

Одновременно можно открыть только один файл.

7.2. Запись данных в файл

7.2.1. *Общее описание*

Оператор-записи-данных-в-файл используется для записи данных на магнитную ленту. Общая синтаксическая форма оператора-записи-данных-в-файл следующая:

PRINT # номер-файла, элемент р . . . р элемент

где номер-файла – это номер, который был присвоен данному файлу.

7.2.2. *Синтаксис*

Оператор-записи-данных-в-файл = PRINT пробел * номер список-вывода

7.2.3. *Примеры*

PRINT#I, A, B, C

7.2.4. *Семантика*

Оператор-записи-данных-в-файл по функционированию полностью совпадает с оператором-вывода (см. п. 3.11) за исключением того, что данные выводятся не на терминал, а записываются на магнитную ленту.

7.2.5. *Исключения*

При записи данных не хватило места на магнитной ленте (неустранимая ошибка).

7.3. О п е р а т о р - ч т е н и я - с - м а г н и т н о й - л е н т ы

7.3.1. *Общее описание*

Оператор-чтения-с-магнитной-ленты позволяет считывать данные с магнитной ленты. Общая синтаксическая форма оператора-чтения-с-магнитной-ленты имеет вид:

INPUT # номер файла, переменная, . . , переменная

7.3.2. *Синтаксис*

Оператор-чтения-с-магнитной-ленты = INPUT пробел * номер список-переменных

7.3.3. *Примеры*

INPUT # I, AI, B¤

7.3.4. *Семантика*

Оператор чтения-с-магнитной-ленты по функционированию полностью совпадает с оператором-ввода (см. п. 3.12.), выполняемому в ракетном режиме.

7.3.6. *Исключения*

Чтение после конца файла, конец магнитной ленты. Неустранимая ошибка. Выводится диагностическое сообщение.

7.4. Ф у н к ц и я - п р и з н а к а - к о н ц а - ф а й л а

7.4.1. *Общее описание*

Эта функция используется для идентификации конца-файла при операции чтения.

7.4.2. *Синтаксис*

Функция-признака-конца-файла = EOF круглая-скобка-левая номер-файла круглая-скобка-правая

7.4.3. *Пример*

IF EOF (# 2) THEN I00

7.4.4. *Семантика*

Функция-признака-конца-файла возвращает нуль (”ложь”), если файл еще не исчерпан. В противном случае она возвращает единицу (”истина”).

7.4.5. *Примечания*

Эту функцию следует использовать в операторе-условного-перехода или в цикле-пока для определения конца файла, открытого для чтения.

7.5. О п е р а т о р - з а к р ы т и я - ф а й л а

7.5.1. *Общее описание*

Оператор-закрытия-файла заканчивает работу с файлом (закрывает файл). Синтаксическая форма оператора-закрытия-файла имеет вид:

CLOSE # номер-файла

7.5.2. *Синтаксис*

Оператор-закрытия-файла = CLOSE пробел* номер номер-файла

7.5.3. *Примеры*

CLOSE 2

7.5.4. *Семантика*

Если файл был открыт для записи, то после выполнения оператора-закрытия-файла файл закрывается, и позже этот файл можно открыть для чтения.

7.5.5. *Примечания*

Если файл был открыт для чтения, то закрывать его необходимо лишь в том случае, когда нужно открыть еще какой-либо файл.

Для файла, открытого для записи, процедура закрытия обязательна.

8. МОДУЛЬ РАБОТЫ С МАГНИТНЫМИ ДИСКАМИ

Данный модуль содержит два уровня и устанавливает синтаксис и семантику операторов для работы с информацией, хранящейся на магнитных дисках.

8.1. **Уровень 1**

Данный уровень описывает средства работы с файлами с последовательным методом доступа.

8.1.1. О п е р а т о р - о т к р ы т и я - ф а й л а

8.1.1.1. *Общее описание*

Оператор-открытия-файла подготавливает (открывает) файл для операций чтения или записи. Общая синтаксическая форма оператора-открытия-файла имеет вид.

OPEN режим, # номер-файла, имя-файла

8.1.1.2. *Синтаксис*

1) оператор-открытия-файла = OPEN кавычки режим кавычки запятая номер номер-файла запятая имя-файла

2) режим = I|O

3) номер-файла = целое

4) имя-файла = текстовая-константа|текстовая-переменная

8.1.1.3. *Примеры*

1) OPEN "I", #I, "TEST.PRN"

8.1.1.4. *Семантика*

Режим I означает, что файл открывается для чтения, т.е. что данные будут читаться с диска в оперативную память.

Режим 0 означает, что данные будут записываться из оперативной памяти на диск.

Количество файлов, одновременно открытых для чтения и записи определяется реализацией, но должно быть не менее двух.

Номер-файла используется для идентификации файлов в операторах ввода/вывода. Номер-файла должен быть целым число в диапазоне от I до числа, определяемого реализацией.

Структура имени-файла определяется реализацией. Имя-файла должно использоваться только в операторе-открытия-файла.

Один и тот же файл может быть одновременно открыт только для чтения или только для записи.

8.1.1.5. *Исключения*

Файл, открываемый для чтения (режим I), не существует на диске (неустранимая ошибка).

8.1.1.6. *Примечания*

Оператор-открытия-файла должен выполняться до любого другого оператора, работающего с магнитным диском.

8.1.2. О п е р а т о р - з а к р ы т и я - ф а й л а

8.1.2.1. *Общее описание*

Оператор-закрытия-файла заканчивает работу с файлом (закрывает файл). Синтаксическая форма оператора-закрытия-файла имеет вид:

CLOSE # номер-файла

8.1.2.2. *Синтаксис*

Оператор-закрытия-файла = CLOSE (номер номер-файла)? (запятая номер-файла)*

8.1.2.3. *Примеры*

CLOSE #1

8.1.2.4. *Семантика*

Если файл был открыт для записи, то после выполнения оператора-закрытия-файла файл закрывается и позже этот файл можно открыть для чтения.

Одним оператором-закрытия-файла можно закрыть сразу несколько файлов. Если список номеров-файлов отсутствует, то закрываются все открытые файлы.

8.1.2.5. *Исключения*

Попытка закрыть неоткрытый файл. Ошибка не является неустранимой. Программа продолжает работу.

8.1.2.6. *Примечания*

Файл, открытый для чтения, необходимо закрывать только в том случае, если его снова нужно читать с начала.

8.1.3. О п е р а т о р ы з а п и с и - д а н н ы х - в - ф а й л и ф о р м а т и р о в а н н о й - з а п и с и - в - ф а й л

8.1.3.1. *Общее описание*

Оператор-записи-данных-в-файл используется для вывода данных в файл на магнитный диск. Его синтаксис и семантика полностью совпадают с оператором-вывода из п. 7.2.

Оператор-форматированной-записи-в-файл используется для форматированного вывода в файл на магнитный диск.

8.1.3.2. *Синтаксис*

Оператор-форматированного-вывода = PRINT пробел номер номер-файла пробел USING текстовая-константа|текстовая-переменная точка-с-запятой список-вывода.

8.1.3.3. *Примеры*

PRINT # 2 USING „# # # #”; A%; B%; C

8.1.3.4. *Семантика*

Операторы записи-данных-в-файл и форматированной-записи-в-файл используются для записи данных в файл на магнитный диск. Данные записываются последовательно в соответствии со списком-вывода.

Семантика оператора-форматированной-записи-в-файл полностью совпадает с семантикой оператора-форматированного-вывода из п. 4.11.

8.1.3.5. *Исключения*

При выполнении операции вывода диск переполнился (неустранимая ошибка).

8.1.4. Операторы чтения-с-магнитного-диска и чтения-текста-с-магнитного-диска

8.1.4.1. *Общее описание*

Операторы чтения-с-магнитного-диска и чтения-текста-с-магнитного-диска используются для чтения данных из файла, хранящегося на магнитном диске.

Синтаксис и семантика оператора-чтения-с-магнитного-диска совпадают с синтаксисом и семантикой оператора-чтения-с-магнитной-ленты, описанного в п. 7.3.

Семантика оператора-чтения-текста-с-магнитного-диска совпадает с семантикой оператора-ввода-текста, описанного в п. 4.1.13, за исключением того, что данные выводятся не с клавиатуры, а из файла.

8.1.4.2. *Синтаксис*

Оператор-чтения-с-магнитного-диска = LINE пробел INPUT номер номер-файла точка-с-запятой текстовая-переменная

8.1.4.3. *Примеры*

LINE INPUT # 3; MI¤

8.1.4.4. *Семантика*

Операторы чтения-с-магнитной-ленты и чтения-текста-с-магнитного-диска используются для ввода данных из файла, хранящегося на магнитном диске. Данные вводятся последовательно в том же порядке, в котором они ранее были записаны в файл.

8.1.4.6. *Исключения*

Чтение после конца файла. Неустранимая ошибка. Выводится диагностическое сообщение.

8.1.5. Функция признака-конца-файла

Синтаксис и семантика функции-признака-конца-файла полностью совпадают с функцией, описанной в п. 7.4.

8.2. **Уровень 2**

Этот уровень описывает средства работы с файлами с произвольным методом доступа.

8.2.1. О п е р а т о р - о т к р ы т и я - ф а й л а

8.2.1.1. *Общее описание*

Оператор-открытия-файла подготавливает (открывает) файл для операций чтения/записи. Общая синтаксическая форма оператора-открытия-файла имеет вид:

OPEN режим, # номер-файла, имя-файла, длина-текста

8.2.1.2. *Синтаксис*

1) оператор-открытия-файла = OPEN кавычки режим кавычки запятая номер номер-файла запятая имя-файла запятая длина-текста

2) режим = R

3) длина-текста = целое

8.2.1.3. *Примеры*

1) OPEN "R", # I, "FILE", 32

8.2.1.4. *Семантика*

При выполнении оператора OPEN для заданного файла назначается буфер ввода/вывода.

Длина-текста – целое выражение, задающее длину текста для файла с произвольным методом доступа. Максимально возможная длина-текста, а также длина-текста по умолчанию (т.е. длина, принимается в том случае, когда она явно не задана в операторе) определяется реализацией.

8.2.1.5. *Примечания*

При работе с файлом с произвольным методом доступа, один раз открыв файл, в него можно записывать и из него можно считывать информацию.

Информация записывается текстами фиксированной длины, указанной в операторе.

8.2.2. В ы д е л е н и е о б л а с т и д л я п е р е м е н н ы х

8.2.2.1 *Общее описание*

Оператор-выделения-области выделяет в буфере файла с последовательным доступом области для переменных.

Синтаксическая форма оператора имеет вид:

FIELD номер-файла, размер-области AS текстовая-переменная. . .

8.2.2.2. *Синтаксис*

1) оператор-выделения-области = FIELD номер-файла запятая размер-поля AS текстовая-переменная (запятая размер-поля AS текстовая-переменная)*

2) размер-поля = целое

8.2.2.3. *Примеры*

1) FIELD # 1, 20 as N ¤

8.2.2.4. *Семантика*

Размер поля – это количество байтов, выделяемое для данной переменной. Общее число байтов, выделяемое оператором-выделения-

области для всех переменных, не должно превосходить длину записи, заданную при открытии файла.

Этот оператор должен быть выполнен до операторов чтения-из-файла и записи-в-файл.

8.2.2.5. *Примечания*

Идентификаторы текстовых-переменных, указанные в операторе-выделения-области, нельзя использовать в левой части оператора-присваивания, а также в качестве переменных в операторе-ввода.

8.2.3. Оператор-чтения-текста

8.2.3.1. *Общее описание*

Этот оператор читает текст из файла с произвольным доступом в буфер. Синтаксическая форма оператора имеет вид:

GET номер-файла, номер-текста

8.2.3.2. *Синтаксис*

1) оператор-чтения-текста = GET номер-файла (запятая номер-текста)?

2) номер-текста = арифметическое-выражение

8.2.3.3. *Примеры*

1) GET # 2, 37I

8.2.3.4. *Семантика*

Этот оператор читает в буфер текст из файла с данным номером.

Если номер-текста опущен, то читается текст со следующим номером. Наибольший возможный номер-текста определяется реализацией.

8.2.4. Оператор-записи-текста

8.2.4.1. *Общее описание*

Этот оператор записывает в файл текст из буфера, Синтаксическая форма оператора имеет вид:

PUT номер-файла, номер-текста

8.2.4.2. *Синтаксис*

Оператор-записи-текста = PUT номер-файла (номер-текста)?

8.2.4.3. *Примеры*

PUT # 4, 20

8.2.4.4. *Семантика*

Этот оператор помещает текст из буфера в файл. Если номер текста опущен, то текст записывается со следующим по порядку номером (после предыдущего PUT). Максимальный номер текста определяется реализацией.

8.2.5. Пересылка данных в буфер

8.2.5.1. *Общее описание*

Операторы-пересылки-данных-в-буфер используются для пересылки данных в буфер файла с произвольным методом доступа (для подготовки к выполнению оператора-записи-текста). Синтаксическая форма этих операторов следующая:

LSET текстовая-переменная-текстовое-выражение

RSET текстовая-переменная-текстовое-выражение

8.2.5.2. *Синтаксис*

1) пересылка-данных-в-буфере-с-выравниванием-влево = LSET текстовая-переменная знак-равенства текстовое-выражение

2) пересылка-данных-в-буфере-с-выравниванием-вправо = RSET текстовая-переменная знак-равенства текстовое-выражение

8.2.5.3. *Примеры*

1) LSET V¤ = NN¤+ "LIST"

8.2.5.4. *Семантика*

Если текстовое выражение требует меньше байтов, чем было отведено для текстовой переменной в операторе-выделения-области, то LSET пересылает выражение в левую часть буфера текста, заполняя правую пробелами, а RSET – в правую часть буфера текста, заполняя левую пробелами. Если же длина текста превышает длину буфера, то самые правые символы теряются. Числа перед пересылкой должны быть преобразованы в текст.

8.2.5.5. *Примечания*

Если при выполнении операторов пересылки буфер не был пуст (ранее не была произведена запись в файл), то старое содержимое буфера теряется.

Идентификаторы текстовых переменных в операторах пересылки должны совпадать с идентификаторами текстовых переменных в операторе-выделения-области.

8.2.6. Функции-преобразования-данных.

8.2.6.1. *Общее описание*

Так как в файлы с произвольным методом доступа можно записывать только текстовую информацию, то для преобразования чисел в тексты и наоборот в реализации должны существовать специальные функции.

8.2.6.2. *Синтаксис*

Функции-преобразования = MKI¤|MKS¤|MKD¤ |CVI|CVS|CVD|

8.2.6.3. *Семантика*

Значения функций-преобразования приведены в табл. 5.

Таблица 5

Функция	Значение функции
CVI (X$)	Преобразует текст в целое число
CVS (X$)	Преобразует текст в число одинарной точности
CVD (X$)	Преобразует текст в число удвоенной точности
MKI$(N%)	Преобразует целое число в текст
MKS$ (X)	Преобразует число одинарной точности в текст
MKD$ (Y #)	Преобразует число удвоенной точности в текст

8.2.6.4. *Примечания*

Размер текста для каждого типа чисел определяется реализацией.

ПРИЛОЖЕНИЕ 1
Справочное

МЕТОД СИНТАКСИЧЕСКОЙ СПЕЦИФИКАЦИИ

Синтаксис, будучи набором синтаксических правил, определяет синтаксические объекты различных типов, такие как программа или выражение, и определяет, какие строки символов являются объектами этих типов.

В синтаксисе прописные буквы, цифры и слова, написанные строчными буквами (возможно, объединенные дефисом), используются как "метаимена", т.е как имена синтаксических объектов. Подавляющее большинство этих метаимен определяется при помощи грамматических правил в терминах других метаимен Для того, чтобы такой процесс был конечным, некоторые метаимена определены как базовые, или "терминальные". Грамматические правила для терминальных метаимен не включены в синтаксис. Все терминальные метаимена, за исключением конструкций, определяемых реализацией, введены в начале и описаны в п. 3.1 Следует отметить, в частности, что все прописные буквы и цифры являются терминальными метаименами, которые определяют их самих.

Кроме того, конструкции, зависящие от реализации, не являются уникальными синтаксическими объектами, но каждое использование такой конструкции определяется для каждой реализации заранее заданным образом. В некоторых случаях в примечаниях соответствующих подразделов приводятся указания по описанию таких объектов.

Метод синтаксической спецификации использует некоторые специальные символы в качестве синтаксических операторов:

1) = – читается, "по определению есть";

2) * – указывает, что объект, стоящий перед ней, может повторяться произвольное число раз (в том числе и ни разу);

3) ? – указывает, что объект, стоящий перед ним, может присутствовать или отсутствовать;

4) | – указывает, что определение справедливо для каждого из объектов, разделенных данным символом;

5) ( ) – указывают на объединение группы объектов в один для символов, описанных в (1) – (3);

6) пробел – используется в синтаксисе для отделения одного объекта от другого.

Если в описании используется несколько синтаксических операторов, то ? и * имеют более высокий приоритет, чем |.

Пример

В п. 3.9 видим следующее: оператор-вызова-подпрограммы = GO пробел * SUB номер-строки.

Это означает, что оператор-вызова-подпрограммы состоит из букв G и O, за которыми следует пробел, *, букв S, U и B номера-строки. Количество пробелов в соответствие с (1) неограничено.

Что такое номер-строки? В п. 3.2 читаем:

номер строки = цифра цифра? цифра? цифра?

Это означает, что номер-строки является цифрой, за которой могут следовать еще от одной до трех цифр.

Что такое цифра? В п. 3.1 читаем:

цифра = 0|1|2|3|4|5|6|7|8|9

Это означает, что цифрой может быть любая из перечисленных арабских цифр.

Ввиду того, что цифры являются терминальными метаименами, т.е. не появляются ни в каком определении слева, разбор синтаксиса закончен.

ПРИЛОЖЕНИЕ 2
Обязательное

ТЕРМИНЫ И ИХ ПОЯСНЕНИЯ

Конец-строки – символ или индикатор, указывающий на завершение строки. Наиболее часто в качестве конца-строки используются символы "возврат каретки" или "возврат каретки" и "перевод строки".

Ошибка – неправильный синтаксис в строке, в результате чего эта строка не может являться частью программы.

Исключение – ситуация, возникающая при выполнении программы, если неверны исходные данные или неверны вычисления или результаты вне диапазонов, предоставляемых ресурсами системы. Если при этом выполнение программы прекращается, то говорят, что данная ситуация привела к неустранимой ошибке.

Идентификатор – набор символов, используемых для идентификации переменной или функции.

Ключевое слово – слова, используемые для идентификации операторов, компонентов операторов или других элементов языка программирования.

Строка – последовательность символов, заканчивающаяся концом-строки.

Машинный минимум – минимальная положительная величина (отличная от нуля), которая может быть обрабтана реализацией Бейсика.

Машинный максимум – положительное или отрицательное значение максимального числа, которое может быть обрабтано данной реализацией Бейсика.

Переполнение – по отношению к арифметическим операциям возникновение условия, когда попытка выполнить действие приводит к результату, превышающему по абсолютной величине машинный максимум.

По отношению к текстовым операциям возникновение условия, когда попытка выполнить действие приводит к результату, содержащему большее количество символов, чем это допускается языковым процессором.

Зона вывода – непрерывная последовательность знакомест в строке вывода, которая может содержать элемент вывода.

Округление – процесс получения числа меньшей точности из числа большей точности с учетом значения опущенной части числа.

Усечение – процесс получения числа меньшей точности путем отбрасывания ненужных младших цифр числа большей точности.

Потеря точности – возникновение ситуации, когда в результате попытки выполнения операций вырабатывается результат, отличный от нуля, но меньший по абсолютной величине машинного минимума.

Графической курсор – в графическом режиме позиция на экране, в которую в данный момент должен осуществляться вывод графической информации.

ПРИЛОЖЕНИЕ 3
Справочное

ТАБЛИЦА ВЫПОЛНЕНИЯ ЛОГИЧЕСКИХ ОПЕРАЦИЙ

X	Y	X AND Y	X OR Y	X XOR Y	X EQU Y	X IMP Y
0	0	0	0	0	I	I
0	I	0	I	I	0	I
I	0	0	I	I	0	0
I	I	I	I	0	I	I

Операция NOT выполняется по правилам:
если X = I, то NOT X = 0 и, если X = 0, то NOT X = I.

ИНФОРМАЦИОННЫЕ ДАННЫЕ

1. **РАЗРАБОТАН И ВНЕСЕН** Академией наук СССР

С.А. Христочевский (руководитель темы), канд. физ.-мат. наук; **М.Л. Гуткин; Н.А. Кучура; В.А. Ермолов; А.Б. Либеров; А.Л. Александров**

2. **УТВЕРЖДЕН И ВВЕДЕН В ДЕЙСТВИЕ** Постановлением Государственного комитета СССР по стандартам от 26.07.88 № 2760

3. **Срок первой проверки – 1992 г.; периодичность проверки – 3 года.**

4. **Стандарт полностью включает международный стандарт ИСО 6373–84 (Е)**

5. **ВВЕДЕН ВПЕРВЫЕ**

6. **ССЫЛОЧНЫЕ НОРМАТИВНО-ТЕХНИЧЕСКИЕ ДОКУМЕНТЫ**

Обозначение НТД, на который дана ссылка	Номер пункта
ГОСТ 27465–87	3.1.1

7. **Переиздание (апрель 1992 г.)**

СОДЕРЖАНИЕ

Редактор *В.М. Лысенкина*
Технический редактор *О.Н. Власова*
Корректор *В.Ф. Малютина*

Подп. в печ. 27.04.92. Формат 60X90 1/16. Бумага офсетная. Гарнитура Пресс-Роман.
Печать офсетная. Усл.-печ. л. 4,5. Усл. кр.-отт. 4,63. Уч.-изд. л. 5,12. Тираж 2125 экз.
Зак. 1265

Ордена "Знак Почета" Издательство стандартов, 123557, Москва, ГСП,
Новопресненский пер., 3

Набрано в Издательстве стандартов на НПУ

Калужская типография стандартов. Калуга, ул. Московская, 256.