2 3056-78
2 3057-78

ГОСУДАРСТВЕННЫЕ СТАНДАРТЫ
СОЮЗА ССР

ЯЗЫКИ ПРОГРАММИРОВАНИЯ

ФОРТРАН
И БАЗИСНЫЙ ФОРТРАН

ГОСТ 23056—78, ГОСТ 23057—78

Издание официальное

ГОСУДАРСТВЕННЫЙ КОМИТЕТ СССР ПО СТАНДАРТАМ
Москва

ГОСУДАРСТВЕННЫЕ СТАНДАРТЫ
СОЮЗА ССР

ЯЗЫКИ ПРОГРАММИРОВАНИЯ

ФОРТРАН
И БАЗИСНЫЙ ФОРТРАН

ГОСТ 23056—78, ГОСТ 23057—78

Издание официальное

МОСКВА—1982

УДК 681.3.065.4:006.354 Группа П85

ГОСУДАРСТВЕННЫЙ СТАНДАРТ СОЮЗА ССР

Язык программирования

ФОРТРАН

Programming language FORTRAN

ГОСТ 23056—78*

Постановлением Государственного комитета стандартов Совета Министров СССР от 7 апреля 1978 г. № 962 срок введения установлен

с 01.01 1979 г.

Настоящий стандарт распространяется на язык программирования ФОРТРАН и устанавливает:

форму представления и правила интерпретации (синтаксис и семантику) программы, записанной на языке ФОРТРАН;

форму представления входных данных, обрабатываемых программой при ее выполнении в автоматизированной системе обработки данных;

форму представления выходных данных, получаемых в результате выполнения программы.

Стандарт не устанавливает:

механизм, которым программа трансформируется для ее выполнения в системе обработки данных (комбинация этого механизма и системы обработки данных называется процессором);

метод передачи программы и ее входных или выходных данных в систему обработки данных и обратно;

действия, необходимые для запуска и управления программой в системе обработки данных;

результаты выполнения программы, если стандарт не устанавливает правил ее интерпретации;

размер и сложность программы;

диапазон или точность представления числовых значений;

состав и форму документации на трансляторы с языка ФОРТРАН и программы, записанные на языке ФОРТРАН.

Стандарт полностью соответствует международной рекомендации ИСО/Р 1539—72.

(Измененная редакция, Изм. № 1).

Издание официальное **Перепечатка воспрещена**

★

* *Переиздание (июнь 1982 г.) с Изменением № 1, утвержденным в июне 1980 г. (ИУС 9—80).*

© *Издательство стандартов, 1982*

1. ОБЩИЕ ПОЛОЖЕНИЯ

1.1. Стандарт предназначен для достижения высокой степени мобильности и машинной независимости программ, записанных на языке ФОРТРАН, позволяющей использовать их в различных автоматизированных системах обработки данных.

1.2. Процессор, выполняющий программы, записанные на языке ФОРТРАН, считается согласованным с настоящим стандартом, если он воспринимает и интерпретирует в соответствии с настоящим описанием по крайней мере те формы и соотношения, которые описаны в настоящем стандарте.

1.3. Любое ограничение или запрет, сформулированные в настоящем стандарте, означает следующее: если в какой-либо программе это ограничение не выполнено или запрет нарушен, то такая программа считается несогласованной с настоящим стандартом.

Разд. 1 **(Измененная редакция, Изм. № 1).**

2. СТРУКТУРА ЯЗЫКА

2.1. Настоящий раздел определяет общую структуру программ, записанных на языке ФОРТРАН. В разделе содержатся также определенные разъяснения, касающиеся смысла некоторых фраз и отдельных слов.

2.2. Программа, которая представляет собой описание некоторой вполне законченной вычислительной процедуры и может быть выполнена соответствующим процессором, называется выполнимой программой (п. 9.1.7). Таким образом, выполнимая программа содержит исчерпывающую информацию о форме записи исходных данных и алгоритмах их переработки с целью получения искомых результатов, о вводимых в употребление внутренних объектах и о форме представления окончательных результатов.

2.2.1. Выполнимая программа состоит из одного или нескольких программных модулей (п. 9.1.3), один (и только один) из которых является головным модулем (п. 9.1.4).

2.2.2. Выполнение программы начинается с выполнения ее головного модуля. Однако в любом программном модуле (в том числе и в головном), за исключением модуля-блока данных, могут использоваться внешние процедуры (разд. 9), описывающие вне данного модуля отдельные процедуры процесса обработки данных. Для определения внешних процедур средствами ФОРТРАНа служат модули-процедуры (разд. 8).

2.3. Внешняя процедура может быть внешней функцией или внешней подпрограммой (разд. 8 и 9). Внешние процедуры могут определяться и другими средствами, отличными от языка ФОРТРАН. Эти средства настоящим стандартом не определяются.

Как отмечалось выше, в выполняемой программе должна содержаться исчерпывающая информация о вводимых в употребление внутренних объектах. В ФОРТРАНе такими объектами являются в частности, общие блоки данных (п. 7.2.1.3). Для придания начальных значений элементам блоков данных служат модули-спецификации (пп. 8.5, 9.1.5). Каждый такой модуль начинается с заголовка спецификации блока данных. В модулях-спецификациях не должны использоваться внешние процедуры.

(Измененная редакция, Изм. № 1).

2.3.1 **(Исключен, Изм. № 1).**

2.4. Каждый программный модуль состоит из предложений и комментариев. В этом смысле головной модуль — это последовательность предложений и комментариев ФОРТРАНа, не содержащая заголовков функций, заголовков подпрограмм и заголовков спецификаций блоков данных; модуль, не являющийся головным, начинается либо с заголовка функции, либо с заголовка подпрограммы, либо с заголовка спецификации блока данных.

2.5. Предложение делится на физические части, называемые строками, первая из которых называется начальной строкой, а остальные — строками-продолжениями. Каждый комментарий представляет собой строку, не являющуюся предложением или его частью (п. 3.2).

2.4,2.5 **(Измененная редакция, Изм. № 1).**

2.5.1. Предложения ФОРТРАНа распадаются на два основных класса: выполняемые (или операторы) и невыполняемые (или объявления). Операторы определяют действия в программе, тогда как объявления (частными случаями которых являются заголовки функций, заголовки подпрограмм и заголовки спецификаций блока данных) описывают способ использования программы, характеристики операндов, способ редактирования данных, вводимые в употребление функции или размещение данных (пп. 7.1, 7.2).

2.6. Синтаксическими элементами предложения являются имена и операции. Имена используются для ссылок на объекты, например, на данные или процедуры. Операции определяют действия над именованными объектами.

2.6.1. Один частный случай имен, имя массива, заслуживает особого рассмотрения. С именем массива должен быть связан размер идентифицируемого массива, определяемый в описании массива (п. 7.2.1.1). Имя массива, дополненное индексом, используется для идентификации конкретного элемента массива (п. 5.1.3).

2.7. Имена данных, арифметические и логические операции, а также операции отношения могут быть связаны в выражения. Выражение служит для задания правил вычисления значения: это значение получается в результате выполнения указанных в выражении операций над именованными данными.

2.8. Для идентификации в ФОРТРАНе используются имена и целые числа без знака (п. 5.1.1). Данные и процедуры именуются. Предложения помечаются целыми числами без знака. Устройства ввода/вывода нумеруются (разд. 3, 6, 7).

2.9. В настоящем стандарте встречаются условные обозначения предложений ФОРТРАНа, содержащие список элементов; во всех таких случаях предполагается, что список содержит по крайней мере один элемент, если не оговорено противное. Например, запись

$$\text{SUBROUTINE } s(a_1, a_2, \ldots, a_n)$$

обозначает заголовок подпрограммы, причем предполагается, что в список, заключенный в круглые скобки, входит по крайней мере одно символическое имя a_i. Таким образом, список элементов есть либо один элемент, либо последовательность элементов, отделенных друг от друга запятой.

Далее предполагается, что множественное число существительного в любой фразе означает в качестве частного случая также и единственное число этого существительного, если только контекст фразы не запрещает такую интерпретацию.

Термин «ссылка» используется со специальным смыслом, определенным в разд. 5.

3. ПРАВИЛА ЗАПИСИ ПРОГРАММЫ

Вводная часть **(Исключена, Изм. № 1).**

3.1. **Алфавит ФОРТРАНа.** При записи программного модуля используются только символы, входящие в алфавит ФОРТРАНа. Этот алфавит делится на три группы символов: цифры, буквы и специальные символы. Множество символов, образующих алфавит ФОРТРАНа, считается неупорядоченным.

3.1.1. **Цифры.** Цифра — это один из десяти символов:

0, 1, 2, 3, 4, 5, 6, 7, 8, 9.

Если не оговорено противное и уместно считать последовательность цифр числом, то оно будет интерпретироваться как число в десятичной системе счисления.

Восьмеричная цифра — это один из восьми символов:

0, 1, 2, 3, 4, 5, 6, 7.

Восьмеричные цифры используются только в операторах останова (п. 7.1.2.7.1) и паузы (п. 7.1.2.7.2).

3.1.2. **Буквы.** Буква — это одна из двадцати шести символов:

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.

3.1.3. **Буквенно-цифровые** символы. Буквенно-цифровой символ — это *либо* буква, *либо* цифра.

3.1.4. **Специальные символы.** Специальный символ — это один из одиннадцати символов:

Символ	Название символа
	Пробел
=	Равно
+	Плюс
—	Минус
*	Звездочка
/	Дробная черта
(	Круглая скобка левая (левая скобка)
)	Круглая скобка правая (правая скобка)
,	Запятая
.	Точка
¤	Знак денежной единицы

3.1.4.1. **Символ пробела.** Символ пробела — это отсутствие какого-либо графического изображения в данной позиции. Кроме специально оговоренных случаев (пп. 3.2.2—3.2.4, 4.2.6, 5.1.1.6, 7.2.3.6 и 7.2.3.8), символ пробела не является значащим и поэтому может свободно использоваться для улучшения наглядности программы в любом ее месте с учетом ограничений на строки продолжения (п. 3.3).

3.2. **Строки.** Строка — это последовательность, состоящая из 72 символов. Каждый символ должен принадлежать алфавиту ФОРТРАНа, за исключением случаев, описанных в пп. 4.2.6, 5.1.1.6, 7.2.3.1 и 7.2.3.8.

Позиции символов в строке последовательно нумеруются слева направо от 1 до 72 включительно.

3.2.1. **Комментарий.** Буква С в позиции 1 какой-либо строки указывает на то, что данная строка является комментарием. За комментарием должны непосредственно следовать либо другой комментарий, либо начальная строка, либо заключительная строка (п. 3.2.2).

Комментарии не оказывают никакого влияния на выполнение программы, их можно использовать для пояснений.

3.2.2. **Заключительная строка.** Заключительной строкой называется такая строка, которая в позициях 1—6 содержит пробелы, а в позициях 7—72 — пробелы и буквы E, N и D. Эти буквы должны следовать в том порядке, в каком они приведены выше, каждая по одному разу, и могут размещаться в любых этих позициях; в остальных позициях должны содержаться пробелы. Заключительная строка указывает процессору конец текста программного модуля (п. 9.1.3). Текст каждого программного модуля обязательно должен завершаться точно одной заключительной строкой.

3.2.3. **Начальная строка.** Начальной строкой называется такая строка, которая не является ни комментарием, ни заключительной строкой и содержит пробел или цифру 0 в позиции 6. В позициях 1—5 содержится либо метка предложения, либо пробелы.

3.2.4. **Строка-продолжение.** Строкой-продолжением называется такая строка, которая не является комментарием и в позиции 6 содержит символ, отличный от пробела и цифры 0.

Строка-продолжение может непосредственно следовать только за начальной строкой или за другой строкой-продолжением.

3.3. **Предложения.** Предложение состоит из одной начальной строки, за которой может следовать до 19 строк-продолжений. Символы, образующие предложение, записываются в позициях 7—72 каждой из строк и считаются упорядоченными: сначала идут символы, записанные в начальной строке, затем — символы, записанные в первой строке-продолжении (если она имеется), затем — символы, записанные во второй строке-продолжении (если она имеется) и т. д. В каждой строке символы считаются упорядоченными по возрастанию номеров позиций, в которых они записаны (учитываются только символы, расположенные в позициях 7—72).

3.4. **Метка предложения.** Любое предложение может быть помечено, чтобы на него можно было ссылаться в других предложениях. Метка предложения состоит из последовательности от одной до пяти цифр. Величина целого без знака, представленного этой последовательностью цифр, не играет роли, но она должна быть больше нуля. Метка предложения должна быть помещена в позициях 1—5 начальной строки этого предложения и может начинаться с любой из этих позиций. В одном программном модуле одной и той же меткой не должно быть помечено более одного предложения. При отождествлении меток ведущие нули не учитываются.

3.5. **Символические имена.** Символическое имя состоит из буквы, за которой может следовать еще до пяти буквенно-цифровых символов (см. пп. 10.1—10.1.10 относительно классификации символических имен и ограничений на их использование).

3.6. **Упорядоченность символов.** Символы, образующие программный модуль, считаются упорядоченными. В частности, любой осмысленный набор символов, образующий имена, строки и предложения, является упорядоченным. Эта упорядоченность определяется упорядоченностью символов в строке (п. 3.2) и порядком следования строк в программном модуле.

3.1, 3.1.1—3.1.4, 3.1.4.1, 3.2, 3.2.1—3.2.4, 3.3—3.6 **(Измененная редакция, Изм. № 1).**

4. ТИПЫ ДАННЫХ

В ФОРТРАНе различают данные шести типов: целые, вещественные, двойной точности, комплексные, логические и текстовые. Каждый тип предназначен для вполне определенных целей и может иметь свое особое внутреннее представление. Поэтому интерпретация операций над данными существенно зависит от типов

этих данных. Тип функции определяет тип того значения, которое доставляется в качестве результата в выражение, содержащее указатель этой функции.

4.1. **Связь данного с его типом.** С именем, используемым для идентификации данного или функции, связывается вполне определенный тип данных. Последовательность символов, образующих константу, определяет как значение, так и тип этой константы.

В каждом программном модуле с символическим именем, представляющим функцию, переменную или массив, связывается только один тип данных. Установленная однажды, эта связь в рамках данного программного модуля должна быть сохранена для любого другого использования этого символического имени, требующего учета типа данных.

Для символического имени тип данных может быть установлен указанием его в объявлении типа (п. 7.2.1.6) для любого типа данных, кроме текстового. Такое явное объявление аннулирует неявную связь, устанавливаемую для целого и вещественного типов (п. 5.3).

Не существует способа установления связи символического имени с текстовым типом данных, поэтому данные этого типа, за исключением констант, идентифицируются с помощью имени одного из остальных типов (см., например, п. 8.4.2.).

(Измененная редакция, Изм. № 1).

4.2. **Свойства данных разных типов.** В следующих разделах определяются математические свойства и свойства представлений каждого из шести типов данных. Значение нуль не считается ни положительным, ни отрицательным для данных типов целый, вещественный и двойной точности.

4.2.1. **Тип целый.** Целое данное — это всегда точное представление целого значения. Оно может принимать только целые (положительные, отрицательные и нулевое) значения.

4.2.2. **Тип вещественный.** Вещественное данное — это процессорное приближение вещественного значения. Оно может принимать положительные, отрицательные и нулевое значения.

4.2.3. **Тип двойной точности.** Данное двойной точности — это процессорное приближение вещественного значения. Оно может принимать положительные, отрицательные и нулевое значения. Точность приближения, не определяемая здесь, должна быть больше, чем для типа вещественный.

4.2.4. **Тип комплексный.** Комплексное данное — это процессорное приближение комплексного значения. Это приближение представлено в виде упорядоченной пары вещественных данных. Первый элемент пары представляет действительную, а второй — мнимую часть комплексного числа. Соответственно, каждый элемент имеет ту же точность приближения, что и вещественное данное.

4.2.5. **Тип логический.** Логическое данное может принимать одно из двух логических значений: «истина» или «ложь».

4.2.6. **Тип текстовый.** Текстовое данное — это последовательность символов. Эта последовательность может состоять из любых символов, допускающих представление в процессоре. В текстовом данном символ пробела является допустимым и значащим.

(Измененная редакция, Изм. № 1).

5. ИДЕНТИФИКАЦИЯ ДАННЫХ И ПРОЦЕДУР

Имена используются как для ссылок на данные и процедуры, так и для любой другой их идентификации.

Термин «ссылка» используется при такой идентификации данного, когда подразумевается, что текущее значение этого данного становится доступным при выполнении предложения (оператора), содержащего эту ссылку. Если данное идентифицируется, но его значение не обязательно становится доступным, то говорят, что данное именуется. Один случай, когда данное именуется, представляет особый интерес — это когда данному присваивается значение и тем самым производится определение или переопределение (значения) данного, т. е. данное либо впервые получает некоторое конкретное значение, либо ему присваивается новое значение. В случае процедуры термин «ссылка» означает, что действия, определяемые этой процедурой, станут доступными при выполнении предложения (оператора), содержащего эту ссылку.

Полное и строгое объяснение понятий «ссылка» и «определение» (включая и «переопределение») содержится в разд. 10.

5.1. **Имена данных и процедур.** С помощью имен данных идентифицируются константы, переменные, массивы или элементы массивов, а также блоки (п. 7.2.1.3). С помощью имен процедур идентифицируются функции и подпрограммы.

5.1.1. **Константы.** Константа является данным, которое всегда определено в процессе выполнения программы и не может быть переопределено (т. е. изменено). Для каждого типа данных имеются свои правила записи констант.

Среди числовых констант (чисел) типа целый, вещественный и двойной точности различаются число без знака и число со знаком. Последнее представляет собой число без знака, непосредственно перед которым расположен знак плюс или минус. Термином «число» обозначается как число со знаком, так и число без знака.

5.1.1.1. **Целое число без знака.** Целое число без знака записывается как непустая последовательность цифр. Значение константы этого вида в точности равно числу, изображаемому в десятичной системе счисления этой константой.

5.1.1.2. **Вещественное число без знака.** Основной формой записи вещественного числа без знака является смешанная дробь, под которой понимается запись вида

P.Q

(где P — целая часть, а Q — дробная часть). Как целая, так и дробная часть есть целое без знака. Одна из этих частей (либо целая, либо дробная) может отсутствовать, т. е. представляться пустой последовательностью цифр. Значение константы этого вида есть процессорное приближение того числа, которое в десятичной системе счисления записывается в виде указанной смешанной дроби.

Десятичная экспонента изображается буквой E, за которой следует целое число (без знака или со знаком). Десятичная экспонента является множителем (применяемым к числу, записанному непосредственно перед десятичной экспонентой), равным приближению результата возведения числа десять в степень, указанную целым числом, записанным после буквы E.

Вещественное число без знака — это либо смешанная дробь, либо смешанная дробь, за которой следует десятичная экспонента, либо целое число без знака, за которым следует десятичная экспонента.

5.1.1.3. **Число двойной точности без знака.** Экспонента двойной точности записывается и трактуется аналогично десятичной экспоненте, за исключением того, что вместо буквы E используется буква D.

Число двойной точности без знака записывается как смешанная дробь, за которой следует экспонента двойной точности, либо как целое число без знака, за которым следует экспонента двойной точности.

5.1.1.4. **Комплексное число.** Комплексное число задается в виде упорядоченной пары вещественных чисел (каждое из которых может быть либо вещественным без знака, либо вещественным со знаком), разделенных запятой и заключенных в скобки. Значением такой константы является приближение комплексного числа, представленного указанной парой чисел, первое из которых представляет действительную, а второе — мнимую часть комплексного числа.

5.1.1.5. **Логическая константа.** Логические константы «истина» и «ложь» записываются как .TRUE. и .FALSE. соответственно.

5.1.1.6. **Текстовая константа.** Запись текстовой константы имеет вид

$$nHh_1h_2...h_n$$

(n — целое без знака ($n>0$); каждое h_i — некоторый символ). Последовательность из n символов, которая следует за буквой H, и образует собственно текстовое данное — константу. После бук-

вы Н могут быть записаны любые n символов, представление которых допустимо в процессоре. В этой последовательности символов, изображающей текстовое данное, символ пробела является значащим. Константа этого типа может встречаться только в списке фактических параметров оператора вызова подпрограммы и в объявлении начальных данных.

(Измененная редакция, Изм. № 1).

5.1.2. **Переменная.** Переменная есть данное, идентифицируемое символическим именем (п. 3.5). На это данное можно ссылаться и его можно определять (т. е. присваивать ему значение).

5.1.3. **Массив.** Массив есть упорядоченный набор данных, имеющий одно, два или три измерения. Массив идентифицируется символическим именем. Идентификация этого упорядоченного набора данных как единого целого достигается посредством использования имени массива.

5.1.3.1. **Элемент массива.** Элементом массива является одна из компонент набора данных, образующего массив. Элемент массива идентифицируется указанием имени массива, непосредственно за которым следует дополнительная конструкция, называемая индексом. Индекс указывает на конкретный элемент массива.

На элемент массива можно ссылаться и его можно определять.

5.1.3.2. **Индекс.** Индекс представляет собой заключенный в скобки список индексных выражений. Если индексных выражений несколько, то они отделяются друг от друга запятой. Число индексных выражений должно соответствовать объявленной размерности массива (п. 7.2.1.1), за исключением вхождения имени элемента массива в объявление эквивалентности (п. 7.2.1.4). Идентифицируемый элемент массива определяется при помощи функции линеаризации (п. 7.2.1.1.1), используя вычисленные значения всех индексных выражений.

5.1.3.3. **Индексные выражения.** Индексное выражение записывается в виде одной из следующих конструкций:

C* V+K
C* V—K
C* V
V+K
V—K
V
K

где C и K — целые без знака, V — ссылка на переменную типа целый (см. разд. 6 относительно правил вычисления выражений; пп. 10.2.8 и 10.3 относительно требований на использование переменной в индексе).

5.1.4. **Процедуры.** Процедура (разд. 8) идентифицируется символическим именем. Процедурой является либо внутренняя функция, либо встроенная функция, либо основная внешняя функция, либо внешняя функция, либо внешняя подпрограмма. Внутренние функции, встроенные функции, основные внешние функции и внешние функции называются общим термином функции (или процедуры-функции), а внешние подпрограммы — термином подпрограммы (или процедуры-подпрограммы).

Функция доставляет результат, который называется значением функции и используется в точке ссылки на эту функцию; подпрограмма этого не делает. Способы ссылки на функции и подпрограммы отличаются друг от друга.

5.2. **Ссылка на функцию.** Ссылка на функцию производится при помощи указателя функции, состоящего из имени функции, за которым следует список фактических параметров, заключенный в скобки. Если список содержит более одного параметра, то они отделяются друг от друга запятой. Допустимые виды фактических параметров функций приведены в разд. 8 (см. п. 10.2.1 относительно требований к ссылкам на функции).

5.3. **Правила типов для идентификаторов данных и процедур.** Тип константы определяется ее изображением. С символическим именем, идентифицирующим блок данных или подпрограмму, не связывается никакой тип.

Тип, который связывается с символическим именем, идентифицирующим переменную, массив или внутреннюю функцию, может быть указан при помощи объявления типа. При отсутствии явного объявления типа с символическим именем связывается тип целый, если первая буква этого имени есть I, J, K, L, M, N; в противном случае связывается тип вещественный.

Если символическое имя встроенной функции или основной внешней функции используется в таком контексте, где оно идентифицирует именно одну из этих функций, то с ним связывается тип соответствующей функции, определенный в табл. 3 и 4.

Если в программном модуле содержатся ссылки на внешнюю функцию, то тип этой функции определяется так же, как и для переменной или массива. Для функции, определенной при помощи модуля-функции, тип определяется либо неявно, по имени функции, либо указывается явно в заголовке функции.

С каждым элементом массива связывается тот же тип, который связан с именем этого массива.

5.4. **Формальные параметры.** Формальный параметр внешней процедуры представляет переменную, массив, подпрограмму или внешнюю функцию.

Если формальный параметр используется в качестве имени внешней функции, то в качестве фактического параметра ему может соответствовать только имя внешней функции (разд. 8).

Если формальный параметр используется в качестве имени внешней подпрограммы, то в качестве фактического параметра ему может соответствовать только имя внешней подпрограммы.

Если формальный параметр используется для ссылки на переменную или на элемент массива, то при задании фактического параметра с этим формальным параметром должно связываться значение того же типа, которое определено для формального параметра по правилам, приведенным в п. 5.3.

Если не оговорено противное, то использование формального параметра в качестве имени переменной, массива или элемента массива допустимо при условии установления надлежащей связи с соответствующим фактическим параметром.

Процесс установления связи фактических параметров с формальными параметрами изложен в разд. 8 и 10.

6. ВЫРАЖЕНИЯ

Настоящий раздел определяет форму представления и правила вычисления арифметических и логических выражений, а также отношений. Отношение употребляется только как компонента логических выражений. Выражение формируется из операндов и знаков операций (см. п. 10.3 об ограничениях на использование операндов в выражениях).

6.1. **Арифметические выражения.** Арифметическое выражение формируется из знаков арифметических операций и арифметических операндов. Как выражение, так и входящие в него операнды идентифицируют значения типа целый, вещественный, двойной точности или комплексный. Знаки арифметических операций:

Знак операции	Представляемая операция
+	Сложение
—	Вычитание
*	Умножение
/	Деление
**	Возведение в степень

Арифметические операции сложения и вычитания могут быть одноместными и двуместными. В случае одноместных операций сложения и вычитания подразумеваемым первым арифметическим операндом является нуль.

Арифметические операнды — это первичное арифметическое выражение, множитель, терм, терм со знаком, простое арифметическое выражение и арифметическое выражение.

Первичное арифметическое выражение — это либо арифметическое выражение, взятое в скобки, либо константа, либо ссыл-

ка на переменную, либо ссылка на элемент массива, либо ссылка на функцию.

Множитель — это либо первичное арифметическое выражение, либо конструкция вида:

первичное арифметическое выражение ** *первичное арифметическое выражение*

Терм — это либо множитель, либо конструкция одного из видов

терм/*множитель*

или

*терм** *множитель*

Терм со знаком — это терм, которому непосредственно предшествует знак + или —.

Простое арифметическое выражение — это либо терм, либо два простых арифметических выражения, разделенные знаком + или —.

Арифметическое выражение — это либо простое арифметическое выражение, либо терм со знаком, либо одна из этих двух конструкций, за которой непосредственно следует знак + или и —, за которым непосредственно следует простое арифметическое выражение.

Первичное арифметическое выражение любого типа может возводиться в степень, показателем которой является первичное арифметическое выражение типа целый; при этом получающийся в результате множитель имеет тот же тип, что и возводимое в степень первичное арифметическое выражение. Первичное арифметическое выражение типа вещественный или двойной точности может возводиться в степень, показателем которой является первичное арифметическое выражение типа вещественный или двойной точности; получающийся в результате множитель имеет тип вещественный, если оба упомянутых первичных арифметических выражения были типа вещественный; в остальных случаях он имеет тип двойной точности. Во всех остальных случаях эффект выполнения операции возведения в степень не определен.

При использовании остальных арифметических операций любой допустимый операнд может сочетаться с любым допустимым операндом того же самого типа; получающийся в результате операнд имеет тот же самый тип. Кроме того, допустимый операнд типа вещественный может сочетаться с допустимым операндом типа двойной точности или комплексный; получающийся в результате операнд имеет соответственно тип двойной точности или комплексный.

6.2. **Отношения.** Отношение состоит из двух арифметических выражений, разделенных знаком операции отношения, и принимает значение «истина» или «ложь» в зависимости от выполнения

или невыполнения этого отношения. Одно из арифметических выражений может быть типа вещественный или двойной точности, тогда другое должно быть также типа вещественный или двойной точности (допустимы все четыре сочетания), либо оба арифметических выражения должны быть типа целый. Если в отношении одно из выражений имеет тип вещественный, а другое — тип двойной точности, то результат будет такой же, как и для отношения, в котором правым арифметическим выражением является нуль двойной точности, левым — разность двух исходных арифметических выражений (в исходном порядке), а знак операции отношения тот же самый. Знаки операций отношения:

Знак операции	Представляемая операция
.LT.	Меньше
.LE.	Меньше или равно
.EQ.	Равно
.NE.	Не равно
.GT.	Больше
.GE.	Больше или равно

6.3. **Логические выражения.** Логическое выражение формируется из знаков логических операций и логических операндов и принимает значения «истина» или «ложь». Знаки логических операций:

Знак операции	Представляемая операция
.OR.	Логическое сложение
.AND.	Логическое умножение
.NOT.	Логическое отрицание

Логические операнды — это первичное логическое выражение, логический множитель, логический терм и логическое выражение.

Первичное логическое выражение — это либо логическое выражение, взятое в скобки, либо отношение, либо логическая константа, либо ссылка на логическую переменную, либо ссылка на элемент логического массива, либо ссылка на логическую функцию.

Логический множитель — это либо первичное логическое выражение, либо знак .NOT., за которым следует первичное логическое выражение.

Логический терм — это либо логический множитель, либо конструкция вида:

логический терм .AND. *логический терм*

Логическое выражение — это либо логический терм, либо конструкция вида:

логическое выражение .OR. *логическое выражение*

6.4. **Вычисление выражений.** Часть выражения нуждается в вычислении только в том случае, если это необходимо для установления значения всего этого выражения. Правила формирования выражений определяют и порядок выполнения операций. Необходимо отметить, что вторым операндом операции вычитания является терм, непосредственно следующий за знаком этой операции. Вычисление выражения может производиться в соответствии с любой правильной последовательностью его формирования с учетом следующих ограничений.

Если два операнда соединены знаком операции, то порядок вычисления этих операндов произвольный. Если математическая операция коммутативна и (или) ассоциативна, то это можно использовать для переупорядочивания операндов при условии сохранения целостности выражений в скобках. Значение множителя или терма типа целый — это ближайшее целое, не превосходящее математического значения, представленного этим множителем или термом. При вычислении термов типа целый, содержащих операцию деления, законы ассоциативности и коммутативности не используются, и, следовательно, вычисление таких термов должно производиться слева направо.

Любое использование имени элемента массива требует вычисления его индекса. Вычисление функции, входящей в выражение, не может изменить значения никакого другого операнда в выражении, операторе присваивания или операторе вызова подпрограммы, содержащих эту ссылку на функцию. Тип выражения, в котором встречается индекс или ссылка на функцию, не влияет (и на него не влияет) на вычисление фактических параметров или индекса.

Не может быть вычислен множитель, требующий возведения первичного арифметического выражения, значение которого отрицательно, в степень, показатель которой имеет тип вещественный или двойной точности. Не может быть вычислен множитель, требующий возведения первичного арифметического выражения, значение которого равно нулю, в степень, значение показателя которой также равно нулю.

Не может быть вычислен никакой операнд, значение которого математически не определено.

7. ПРЕДЛОЖЕНИЯ

Предложения ФОРТРАНа можно разделить на выполняемые и невыполняемые. Выполняемые предложения — операторы — определяют действия; невыполняемые предложения — объявле-

ния — описывают характеристики и упорядочение данных, способ редактирования данных, вводимые в употребление функции и классификацию программных модулей.

7.1. **Операторы.** Имеется три типа операторов:

операторы присваивания;

операторы управления;

операторы ввода/вывода.

7.1.1. **Операторы присваивания.** Существует три типа операторов присваивания:

арифметический оператор присваивания;

логический оператор присваивания;

оператор предписания.

7.1.1.1. **Арифметический оператор присваивания.** Арифметический оператор присваивания имеет вид

$$v = e$$

(v — имя переменной или имя элемента массива любого типа, отличного от логического;

e — арифметическое выражение).

Выполнение такого оператора заключается в вычислении выражения e и изменении значения v в соответствии с табл. 1.

Таблица 1

Тип v	Тип e	Действие
Целый	Целый	Присв.
То же	Вещественный	Фикс.; Присв.
»	Двойной точности	Фикс.; Присв.
»	Комплексный	Н
Вещественный	Целый	Плав.; Присв.
То же	Вещественный	Присв.
»	Двойной точности	Дв.; Вещ. присв.
»	Комплексный	Н
Двойной точности	Целый	Дв. плав.; Присв.
То же	Вещественный	Дв.; Присв.
»	Двойной точности	Присв.
»	Комплексный	Н
Комплексный	Целый	Н
То же	Вещественный	Н
»	Двойной точности	Н
»	Комплексный	Присв.

Примечания:

1. «Н» — недопустимая комбинация.
2. «Присв.» — передача результирующего значения без изменений.
3. «Вещ. присв».— передача результирующего значения как вещественного данного с максимальной степенью точности.
4. «Дв.» — вычисление выражения в соответствии с правилами, изложенными в п. 6.1 (или более точными), и затем «Дв. плав».
5. «Фикс.» — отбрасывание дробной части результата и преобразование полученного значения в форму целого данного.
6. «Плав.» — преобразование значения в форму вещественного данного.
7. «Дв. плав.» — преобразование значения в форму данного двойной точности с максимальной степенью точности.

(Измененная редакция, Изм. № 1).

7.1.1.2. **Логический оператор присваивания.** Логический оператор присваивания имеет вид

v = e

(v — имя логической переменной или имя элемента логического массива;
e — логическое выражение).

Выполнение такого оператора заключается в вычислении логического выражения e и присваивании вычисленного значения логическому объекту v.

7.1.1.3. **Оператор предписания.** Оператор предписания имеет вид

ASSIGN k TO i

(k — метка оператора;
i — имя переменной типа целый).

После выполнения такого оператора последующее выполнение любого оператора перехода по предписанию (п. 7.1.2.1.2), использующего эту переменную, приведет к тому, что следующим (п. 9.2) будет выполняться оператор, помеченный предписанной меткой k, при условии, что к этому времени не было переопределения переменной i. Меткой k должен быть помечен оператор в том же программном модуле, в котором встречается этот оператор предписания.

После того, как переменная типа целый использована в операторе предписания, на нее нельзя ссылаться ни в каком предложении, кроме оператора перехода по предписанию, до тех пор, пока она не будет переопределена (п. 10.2.3).

7.1.2. **Операторы управления.** Существует восемь типов операторов управления:

операторы перехода;
условный арифметический оператор;
условный логический оператор;

НАУЧНО-ТЕХНИЧЕСКАЯ БИБЛИОТЕКА ВНИИСТ

оператор вызова подпрограммы;
оператор возврата;
оператор продолжения;
операторы останова и паузы;
оператор цикла.

Метки, используемые в операторах управления, должны помечать операторы в том же программном модуле, в котором используются эти операторы управления.

7.1.1.3, 7.1.2. **(Измененная редакция, Изм. № 1).**

7.1.2.1. **Операторы перехода.** Существует три типа операторов перехода:

безусловный оператор перехода;
оператор перехода по предписанию;
вычисляемый оператор перехода.

7.1.2.1.1. **Безусловный оператор перехода.** Безусловный оператор перехода имеет вид

GOTO k

(k — метка оператора).

Результат выполнения этого оператора состоит в том, что следующим будет выполняться оператор, помеченный этой меткой k.

7.1.2.1.2. **Оператор перехода по предписанию.** Оператор перехода по предписанию имеет вид

GOTO i, (k_1,k_2,....,k_n)

(i — имя переменной типа целый;
каждое k_j — метка оператора).

К моменту выполнения оператора перехода по предписанию переменной i должно быть присвоено текущее значение предшествующим выполнением оператора предписания (п. 7.1.1.3); этим значением должна быть одна из меток списка, заключенного в скобки. Результат выполнения такого оператора перехода состоит в том, что следующим будет выполняться оператор, помеченный этой меткой.

7.1.2.1.3. **Вычисляемый оператор перехода.** Вычисляемый оператор перехода имеет вид

GOTO (k_1, k_2,..., k_n), i

(каждое k_j — метка оператора;
i — имя переменной типа целый).

Результат выполнения этого оператора состоит в том, что следующим будет выполняться оператор, помеченный меткой k_{i_0},

где i_0 — значение переменной i к моменту выполнения данного оператора перехода. Действие этого оператора определено только для i_0, удовлетворяющих условию $1 \leqslant i_0 \leqslant n$ (см. пп. 10.2.8. и 10.3 относительно использования переменной типа целый в вычисляемом операторе перехода).

7.1.2.2. **Условный арифметический оператор.** Условный арифметический оператор имеет вид

IF (e) k_1, k_2, k_3

(e — арифметическое выражение типа целый, вещественный или двойной точности;
каждое k_i — метка оператора).

Условный арифметический оператор служит для разветвления вычислительного процесса по трем возможным путям. При выполнении этого оператора сначала вычисляется выражение e, после чего в качестве следующего выполняется оператор, помеченный меткой k_1, k_2 или k_3 при значении e меньшем нуля, равном нулю или большем нуля соответственно.

7.1.2.3. **Условный логический оператор.** Условный логический оператор имеет вид

IF(e) S

(e — логическое выражение;
S — любой оператор, кроме оператора цикла и условного логического оператора).

При выполнении этого оператора сначала вычисляется логическое выражение e. Если e принимает значение «истина», то выполняется оператор S. Если e принимает значение «ложь», то оператор S выполняется так, как если бы он был оператором продолжения (п. 7.1.2.6) (т. е. в этом случае оператор S фактически не выполняется).

7.1.2.4. **Оператор вызова подпрограммы.** Оператор вызова подпрограммы имеет вид

CALL s(a_1, a_2,..., a_n)

или

CALL s

(s — имя подпрограммы;
каждое a_i — фактический параметр (п. 8.4.2)).

В начале выполнения оператора вызова подпрограммы происходит обращение к указанному модулю-подпрограмме. Возврат управления из этого модуля завершает выполнение оператора вызова подпрограммы.

(Измененная редакция, Изм. № 1).

7.1.2.5. **Оператор возврата.** Оператор возврата имеет вид

RETURN

Оператор возврата используется только в модуле-процедуре и предназначен для того, чтобы отмечать его логический конец.

Если этот оператор используется в модуле-подпрограмме, то результат его выполнения состоит в возврате управления в тот модуль, из которого было произведено обращение к рассматриваемому модулю-подпрограмме.

Если этот оператор используется в модуле-функции, то результат его выполнения состоит в возврате управления в тот программный модуль, из которого было произведено обращение к рассматриваемому модулю-функции, и в этот момент становится доступным значение функции (п. 8.3.1), определенное этим модулем.

7.1.2.6. **Оператор продолжения.** Оператор продолжения имеет вид

CONTINUE

В результате выполнения этого оператора просто продолжается нормальный порядок выполнения операторов, т. е. этот оператор не вызывает никаких иных действий.

7.1.2.7. **Операторы останова и паузы.** Существует два вида операторов:

оператор останова;

оператор паузы.

(Измененная редакция, Изм. № 1).

7.1.2.7.1. **Оператор останова.** Оператор останова имеет вид

STOP n

или

STOP

(n — последовательность от одной до пяти восьмеричных цифр).

В результате выполнения этого оператора завершается выполнение программы.

7.1.2.7.2. **Оператор паузы.** Оператор паузы имеет вид

PAUSE n

или

PAUSE

(n — последовательность от одной до пяти восьмеричных цифр).

Выполнение этого оператора состоит из двух этапов. В результате выполнения первого из них происходит приостановка выполнения программы. На время этой приостановки становится доступной последовательность восьмеричных цифр п. Для возобновления выполнения программы необходимы действия, внешние по отношению к ней. Если выполнение возобновляется без каких-либо изменений состояния процессора, то выполняется второй этап оператора паузы, в результате чего продолжается нормальный порядок выполнения операторов (п. 9.2).

7.1.2.8. **Оператор цикла**

7.1.2.8.1. Оператор цикла имеет вид

$$DO\ n\ i = m_1,\ m_2,\ m_3$$

или

$$DO\ n\ i = m_1,\ m_2$$

(n — метка оператора;
i — имя переменной типа целый;
каждое m_j — целое без знака либо имя переменной типа целый).

Оператор, помеченный меткой n и называемый закрывающим оператором тела цикла, должен находиться в том же программном модуле, что и рассматриваемый оператор цикла, и физически должен помещаться после него. Закрывающий оператор не может быть оператором перехода, возврата, останова, паузы, цикла, условным арифметическим оператором, а также условным логическим оператором, содержащим какой-либо из указанных здесь операторов. Переменная i называется управляющей переменной; m_1 называется начальным параметром, m_2 — конечным параметром и m_3 — параметром приращения. При использовании оператора цикла второго вида, в котором m_3 явно не указывается, считается, что параметр приращения есть единица. Во время выполнения оператора цикла значения m_1, m_2 и m_3 должны быть больше нуля.

Под телом оператора цикла понимается последовательность операторов, начиная с первого по порядку оператора, следующего за рассматриваемым оператором цикла, и кончая его закрывающим оператором. В частности, если тело одного оператора цикла содержит другой оператор цикла, то тело этого другого оператора цикла должно быть подмножеством тела первого.

Правильным гнездом называется множество операторов цикла и их тел — таких, что первый встречающийся закрывающий оператор какого-либо из этих операторов цикла физически следует за последним встречающимся оператором цикла из этого множества (этот оператор цикла называется самым внутренним), а первый встречающийся оператор цикла из этого множества не входит в тело никакого другого оператора цикла. В теле самого

внутреннего оператора цикла не может встречаться оператор цикла.

7.1.2.8.2. Оператор цикла служит для задания цикла в программе. Действия, порождаемые выполнением оператора цикла, описываются следующими шестью шагами:

а) управляющей переменной присваивается значение, представленное начальным параметром; это значение не должно превышать значения, представленного конечным параметром;

б) выполняется тело оператора цикла;

в) если управление достигает закрывающего оператора, то после его выполнения управляющая переменная того оператора цикла, который начал выполняться позже всех и тело которого заканчивается этим закрывающим оператором, увеличивается на значение, представленное соответствующим параметром приращения;

г) если значение управляющей переменной, полученное в результате выполнения шага, указанного в подпункте *в*, не превышает значения, представленного соответствующим конечным параметром, то повторяются описанные выше действия, начиная с указанных в подпункте *б*, с учетом того, что под телом цикла, о котором идет речь, понимается тело того оператора цикла, управляющая переменная которого позже всех получила приращение. Если же значение управляющей переменной оказалось больше значения, представленного соответствующим конечным параметром, то оператор цикла считается завершенным и значение его управляющей переменной становится неопределенным;

д) если имеются другие операторы цикла, тела которых заканчиваются упомянутым закрывающим оператором, то значение управляющей переменной того из этих операторов цикла, выполнение которого началось позже всех, увеличивается на значение, представленное соответствующим параметром приращения, и повторяются действия, указанные в подпункте *г*, до тех пор, пока не будут завершены все операторы цикла, тела которых заканчиваются упомянутым закрывающим оператором. После этого выполняется оператор, следующий за этим закрывающим оператором.

В оставшейся части данного пункта (п. 7.1.2.8) под оператором перехода или условным арифметическим оператором понимается также и условный логический оператор, содержащий соответственно оператор перехода или условный арифметический оператор;

е) после выхода из тела оператора цикла в результате выполнения оператора перехода или условного арифметического оператора, т. е. способом, отличным от завершения оператора цикла, значение управляющей переменной этого оператора цикла опре-

делено и равно последнему ее значению, достигнутому по правилам предыдущих шагов.

7.1.2.8.3. Считается, что оператор цикла имеет расширенное тело, если выполнены следующие условия:

а) в правильном гнезде внутри тела самого внутреннего оператора цикла имеется оператор перехода или условный арифметический оператор, который может передать управление вовне этого гнезда;

б) вне гнезда имеется оператор перехода или условный арифметический оператор, который с учетом всех возможных последовательностей выполнения операторов в данном программном модуле может быть выполнен после оператора, указанного в подпункте *а*, и его выполнение может привести к возврату управления в тело самого внутреннего оператора цикла того же правильного гнезда.

Если оба эти условия выполнены, то расширенное тело определяется как тело оператора цикла вместе с его расширением, т. е. множеством всех операторов, которые могут быть выполнены между всеми парами операторов, передающих управление, первый из которых удовлетворяет условию, указанному в подпункте *а*, а второй — условию, указанному в подпункте *б*. Первый оператор пары не включается в расширение тела, а второй — включается. Оператор перехода или условный арифметический оператор не могут приводить к передаче управления вовнутрь тела оператора цикла, за исключением того случая, когда эти операторы выполняются как часть расширенного тела данного оператора цикла. Кроме того, расширение тела оператора цикла не может содержать оператор цикла (того же самого программного модуля), который в свою очередь имеет расширенное тело. Если в теле оператора цикла встречается обращение к процедуре, то действия, определяемые этой процедурой, считаются временно, т. е. на время выполнения этой процедуры, включенными в тело оператора цикла.

7.1.2.8.4. Управляющая переменная, начальный и конечный параметры, а также параметр приращения оператора цикла не могут переопределяться при выполнении тела или расширенного тела этого оператора цикла.

Если оператор является закрывающим более чем для одного оператора цикла, то метка этого закрывающего оператора не может использоваться ни в каком операторе перехода или в условном арифметическом операторе, за исключением случая, когда такой оператор встречается в теле самого внутреннего оператора цикла с этим закрывающим оператором.

7.1.3. **Операторы ввода/вывода.** Существует два типа операторов ввода/вывода:

основные операторы ввода/вывода;

вспомогательные операторы ввода/вывода.

К первому типу относятся операторы, в результате выполнения которых передаются записи из последовательного файла во внутреннюю память и обратно. Ко второму типу относятся операторы перемотки и сдвига назад, назначение которых состоит в установке файла в определенную позицию, а также оператор разметки файла, осуществляющий вывод специальной записи «конец файла».

В дальнейшем считается, что u и f обозначают соответственно устройство ввода/вывода и спецификацию формата. Устройство вода/вывода идентифицируется значением типа целый, поэтому u может быть либо целым без знака, либо именем переменной типа целый, значение которой и идентифицирует определенное устройство. Спецификация формата описана в п. 7.2.3. В свою очередь, f может быть меткой объявления формата или именем массива; в случае метки соответствующее объявление формата должно находиться в том же программном модуле, что и оператор ввода/вывода, использующий эту метку; в случае имени массива должны быть выполнены требования из п. 7.2.3.10.

7.1.2.8.1—7.1.2.8.4, 7.1.3. **(Измененная редакция, Изм. № 1).**

7.1.3.1. **Свойства устройств ввода/вывода.** Считается, что с конкретным устройством ввода/вывода связан только один последовательный файл. Такое устройство обладает следующими свойствами:

а) если устройство содержит записи, то они считаются упорядоченными;

б) существует единственная позиция устройства, называемая начальной. Если устройство не содержит записей, то оно находится в начальной позиции. Если устройство находится в начальной позиции и содержит записи, то первая запись устройства считается очередной записью;

в) если устройство находится в позиции, не являющейся начальной, то существует единственная предыдущая запись, связанная с этой позицией. Наименьшая из всех записей в смысле порядка, подразумеваемого в подпункте *а*, следующая за этой предыдущей записью, считается очередной записью;

г) по завершении выполнения оператора вывода или оператора разметки не существует записи, следующей за записью, образованной этим оператором;

д) при любой передаче очередной записи позиция устройства изменяется так, что эта очередная запись становится предыдущей записью.

Если какое-либо устройство не обладает каким-то из перечисленных выше свойств, то некоторые из определенных ниже опе-

раторов не могут содержать ссылку на это устройство; действие таких операторов для этого устройства не определено.

7.1.3.2. **Основные операторы ввода/вывода.** Основные операторы ввода/вывода служат для передачи записей. Каждый такой оператор может содержать список имен переменных, массивов и элементов массивов. При вводе именованным элементам присваиваются значения, а при выводе их значения передаются вовне.

Записи могут быть форматными и бесформатными. Форматная запись состоит из последовательности символов, допускающих представление в процессоре. Передача такой записи требует ссылки на спецификацию формата, которая определяет необходимые преобразования и размещение записи (п. 7.2.3). Число записей, передаваемых при выполнении операторов форматного чтения или записи, зависит от списка ввода/вывода и указанной спецификации формата (п. 7.2.3.4). Бесформатная запись состоит из последовательности значений в виде, определяемом процессором. Когда выполняется оператор форматного или бесформатного ввода, требуемые записи на указанном устройстве должны быть соответственно форматными или бесформатными.

(Измененная редакция, Изм. № 1).

7.1.3.2.1. **Списки ввода/вывода.** Список ввода определяет имена переменных и элементов массивов, которым присваиваются значения при вводе. Список вывода определяет те переменные и элементы массивов, значения которых передаются при выводе вовне. Списки ввода и списки вывода устроены одинаково.

Списком называется либо простой список, либо простой список, заключенный в круглые скобки, либо список с циклом, либо два списка, разделенных запятой.

Простой список есть либо имя переменной, либо имя элемента массива, либо имя массива, либо два простых списка, разделенных запятой.

Список с циклом — это взятая в круглые скобки последовательность, состоящая из списка и спецификации цикла, разделенных запятой.

Спецификация цикла имеет вид

$$i=m_1, m_2, m_3$$

или

$$i=m_1, m_2$$

(i, m_1, m_2 и m_3 определяются так же, как и для оператора цикла (п. 7.1.2.8)).

Область действия спецификации цикла — это список, входящий в состав списка с циклом; для списков ввода i, m_1, m_2 и m_3 могут встречаться внутри этой области только в индексах.

Имя переменной или имя элемента массива в списке задают самих себя. Имя массива задает имена всех элементов массива,

определенных описанием массива, и эти элементы задаются в порядке, определяемом функцией линеаризации (п. 7.2.1.1.1).

Элементы списка считаются упорядоченными в соответствии с их вхождением в список при его просмотре слева направо. Это упорядочение элементов в списке с циклом имеет место для каждого очередного повторения цикла.

7.1.3.2.2. **Оператор форматного ввода.** Оператор форматного ввода имеет вид

READ (u,f) k

или

READ (u,f)

(k — список ввода).

В результате выполнения этого оператора вводятся очередные записи с устройства, заданного u. Вводимые данные просматриваются и преобразуются в соответствии с форматом, заданным f. Полученные в результате значения присваиваются элементам, определенным списком k (см. п. 7.2.3.4).

7.1.3.2.3. **Оператор форматного вывода.** Оператор форматного вывода имеет вид

WRITE (u,f) k

или

WRITE (u,f)

(k — список вывода).

В результате выполнения этого оператора создаются очередные записи на устройстве, заданном u. Список вывода k определяет последовательность передаваемых значений. Эти значения преобразуются и разносятся по позициям в соответствии с форматом, заданным f (см. п. 7.2.3.4).

7.1.3.2.4. **Оператор бесформатного ввода.** Оператор бесформатного ввода имеет вид

READ (u) k

или

READ (u)

(k — список ввода).

В результате выполнения этого оператора с устройства, заданного u, вводится очередная запись и — если имеется список ввода — значения, содержащиеся в этой записи, последовательно присваиваются элементам, определенным списком k. Последовательность значений, требуемая списком, не может быть длиннее последовательности значений в бесформатной записи.

7.1.3.2.5. **Оператор бесформатного вывода.** Оператор бесформатного вывода имеет вид

WRITE (u) k

(k — список вывода).

В результате выполнения этого оператора на устройстве, заданном u, создается очередная запись, состоящая из последовательности значений, определяемой списком k.

7.1.3.3. **Вспомогательные операторы ввода/вывода.** Существует три типа вспомогательных операторов ввода/вывода:

оператор перемотки;

оператор сдвига назад;

оператор разметки.

7.1.3.3.1. **Оператор перемотки.** Оператор перемотки имеет вид

REWIND u

В результате выполнения этого оператора устройство, заданное u, устанавливается в начальную позицию.

7.1.3.3.2. **Оператор сдвига назад.** Оператор сдвига назад имеет вид:

BACKSPACE u

Если устройство, заданное u, находится в начальной позиции, то в результате выполнения этого оператора просто продолжается нормальный порядок выполнения операторов; в противном случае, кроме этого, позиция этого устройства изменяется таким образом, что запись, которая до выполнения этого оператора была предыдущей, становится очередной.

7.1.3.3.3. **Оператор разметки.** Оператор разметки имеет вид:

ENDFILE u

В результате выполнения этого оператора на устройство, заданное u, выводится специальная запись «конец файла». «Конец файла» — это единственная запись, обозначающая границу последовательного файла. Если запись «конец файла» встретится при выполнении какого-либо оператора ввода, то действие такого оператора не определено.

7.1.3.4. **Вывод форматных записей на печать.** При передаче форматной записи на печать ее первый символ не печатается, а определяет продвижение по вертикали следующим образом.

Символ	Продвижение по вертикали перед печатью
Пробел	Одна строка
0	Две строки
1	К первой строке следующей страницы
+	Никакого продвижения

7.2. **Объявления.** Имеется пять типов объявлений:

объявления спецификаций;

объявление начальных данных;

объявление формата;
объявление внутренней функции;
заголовки (функций, подпрограмм, спецификаций блоков данных).

Ограничения на использование символических имен в объявлениях изложены в п. 10.2.

Объявления внутренних функций и заголовки (функций, подпрограмм, спецификаций блоков данных) рассматриваются в разд. 8.

7.1.3.4, 7.2. **(Измененная редакция, Изм. № 1).**

7.2.1. **Объявления спецификаций.** Имеется пять типов объявлений спецификаций:
объявление массивов;
объявление общих объектов;
объявление эквивалентности;
объявление внешних имен;
объявление типа.

7.2.1.1. **Описание массива.** Описание массива задает характеристики массива, используемого в рассматриваемом модуле.

Описание массива указывает символическое имя массива, число измерений (одно, два или три) и размеры по каждому измерению. Описание массива может встречаться в объявлениях типа, массивов или общих объектов.

Описание массива имеет вид

v(i)

(v — символическое имя, называемое именем описания;
i — список границ).

Список границ состоит из одного, двух или трех выражений (верхних границ), каждое из которых может быть не равным нулю целым без знака или именем отличной от нуля переменной типа целый. Если список границ состоит более чем из одного выражения, то они отделяются друг от друга запятой. Если i не содержит ни одной переменной, то i называется постоянным списком границ.

Наличие списка границ в каком-либо описании служит для информирования процессора о том, что это имя описания является именем массива. Число выражений, образующих список границ, указывает размерность массива. Значения выражений в описании массива определяют максимальное значение, которое может принимать индекс в любом имени элемента этого массива (п. 7.2.1.1.1).

Значение индекса в имени элемента массива не должно быть меньше единицы или больше максимального значения, определенного описанием этого массива.

7.2.1.1.1. **Функция линеаризации массива и значение индекса.** В табл. 2 для заданных размерности, списка границ и индекса в имени элемента массива приведено значение индекса этого элемента массива, а также максимальное значение, которое могут принимать индексы в именах элементов этого массива. Значения всех индексных выражений должны быть больше нуля.

Функция линеаризации упорядочивает все элементы любого массива. Значение этой функции для некоторого данного элемента получается прибавлением единицы к соответствующему значению, указанному в графе «Значение индекса». Элемент массива, индекс которого имеет это значение, следует непосредственно за данным элементом. Последний элемент массива — это элемент, значение индекса которого равно максимально допустимому значению; для этого элемента не существует непосредственно следующего за ним элемента.

Таблица 2

Размерность	Список границ	Индекс	Значение индекса	Максимальное значение индекса
1	(A)	(a)	a	A
2	(A, B)	(a, b)	$a+A \cdot (b-1)$	$A \cdot B$
3	(A, B, C)	(a, b, c)	$a+A \cdot (b-1)+$ $+A \cdot B \cdot (c-1)$	$A \cdot B \cdot C$

Примечание. a, b и c — индексные выражения.
A, B и C — верхние границы по измерениям.

7.2.1.1.2. **Регулируемые размеры.** Если какое-либо из выражений, входящих в список границ, является именем переменной, то описываемый массив называется массивом с регулируемыми размерами, а имена переменных в списке границ — регулируемыми размерами. Такой массив допустим только в модуле-процедуре. Список формальных параметров такого модуля-процедуры в этом случае должен содержать имя массива и имена переменных типа целый, представляющих регулируемые размеры. Значения фактических параметров, которые представляют размеры массива в списке фактических параметров при ссылке на эту процедуру, должны быть определены (п. 10.2) до обращения к соответствующему модулю-процедуре; эти значения не могут быть изменены или стать неопределенными в процессе выполнения этого модуля-процедуры. Размер фактического массива не может быть превышен. Для каждого формального параметра-массива, появляющегося в выполнимой программе (п. 9.1.6), должно существовать по крайней мере одно описание массива с постоянным списком границ, связанное с упомянутым массивом через обращения к модулям-процедурам.

Если символическое имя фигурирует в объявлении общих объектов модуля-процедуры, то оно не может идентифицировать массив с регулируемыми размерами.

7.2.1.2. **Объявление массивов.** Объявление массивов имеет вид

$$\text{DIMENSION } v_1(i_1),\ v_2(i_2),\ldots,v_n(i_n)$$

(каждое v_j (i_j) — описание массива).

7.2.1.3. **Объявление общих объектов.** Объявление общих объектов имеет вид

$$\text{COMMON } /x_1/a_1 \ \ldots\ /x_n/a_n$$

(каждое a_i — непустой список имен переменных, имен массивов или описаний массивов (в этом списке не должны встречаться формальные параметры);

каждое x_i — символическое имя либо пусто).

Если x_1 пусто, то первые две дробные черты необязательны. Каждое x_i, называемое именем блока, не имеет никакого отношения к каким-либо переменным или массивам с тем же самым символическим именем как в том программном модуле, в котором встречается это объявление общих объектов, так и в любом другом (см. п. 10.1.1 относительно ограничений на использование имен блоков).

Все объекты, указанные между x_i и следующим именем блока (или до конца объявления, если за x_i больше не встречается ни одного имени блока), считаются входящими в общий блок с именем x_i. Все объекты, указанные в начале объявления до первого появления имени блока, или все без исключения указанные объекты, если в объявлении нет ни одного имени блока, считаются входящими в непомеченный общий блок. Наличие двух дробных черт, между которыми не содержится имени блока, также означает, что следующие за этими дробными чертами объекты входят в непомеченный общий блок.

Одно и то же имя общего блока может встречаться более одного раза как в одном объявлении общих объектов, так и в программном модуле. Все объекты, связанные таким образом с одним и тем же общим блоком, располагаются в нем в порядке их появления (п. 10.1.2). При этом первый элемент массива будет непосредственно следовать за предыдущим объектом (если таковой существует), а последний элемент массива будет непосредственно предшествовать следующему объекту (если таковой существует).

Размер общего блока в программном модуле равен сумме объемов памяти, требуемых для размещения объектов, включенных в общий блок при помощи объявлений общих объектов и эквивалентности (п. 7.2.1.4). Размеры общих блоков с одним и тем же именем во всех программных модулях, входящих в выпол-

нимую программу, должны совпадать. Однако размер непомеченного общего блока не обязан быть одним и тем же в разных совместно выполняемых программных модулях. Размер блока измеряется в единицах памяти (п. 7.2.1.3.1).

7.2.1.3.1. **Соответствие общих блоков.** Пусть все объявления общего блока с данным именем в программных модулях выполнимой программы определяют одну и ту же длину общего блока и, кроме того, типы объектов, расположенных в одинаковых позициях различных объявлений этого блока, совпадают. Тогда значения в соответствующих позициях общего блока (определенных по числу предшествующих единиц памяти) будут во всей выполнимой программе одними и теми же.

Считается, что объект типа двойной точности или типа комплексный занимает две последовательные единицы памяти, а объект типа целый, вещественный и логический — одну единицу памяти.

Для общих блоков с одним и тем же именем или для непомеченного общего блока:

во всех тех программных модулях, где в данной позиции (определяемой числом предшествующих единиц памяти) задан один и тот же тип, ссылки на эту позицию дают одно и то же значение;

правильная ссылка, сделанная на конкретную позицию, подразумевает тип, заданный в объявлениях, если последнее по времени присваивание в эту позицию было того же типа.

7.2.1.4. **Объявление эквивалентности.** Объявление эквивалентности имеет вид

$$\text{EQUIVALENCE } (k_1), (k_2),\ldots,(k_n)$$

(каждое k_1 — список вида $a_1, a_2,\ldots, a_{m_i}$

каждое a_1 — имя переменной или имя элемента массива, индекс которого содержит только константы;

$m_i \geqslant 2$).

В списки k не могут входить формальные параметры. Число индексных выражений в имени элемента массива должно либо совпадать с числом измерений в описании массива, либо должно быть равно единице (функция линеаризации определяет правило, по которому любой массив можно свести к одномерному массиву той же самой длины).

Все элементы, образующие какой-либо список k_1 в объявлении эквивалентности, размещаются в памяти, начиная с одной и той же единицы памяти. Объявление эквивалентности не должно использоваться для того, чтобы сделать две или более величины математически эквивалентными.

Отведение памяти для переменных или массивов, объявленных непосредственно в объявлении общих объектов, производится

только с учетом их типов, объявлений общих объектов и описаний массивов. Объявленным таким образом объектам всегда отводится память в том порядке, в котором они следуют в объявлении общих объектов.

Результатом объявления эквивалентности по отношению к общим объектам может быть лишь удлинение общего блока; при этом разрешается только такое удлинение, которое расширяет общий блок за последний, но не за первый объект этого блока, определенный непосредственно в объявлении общих объектов.

Если две переменные или элементы двух массивов совмещаются в памяти в результате эффекта объявления эквивалентности, то имена этих переменных или массивов в данном программном модуле не могут одновременно встречаться в объявлении общих объектов.

Информация, содержащаяся в пп. 7.2.1.1.1, 7.2.1.3.1 и в настоящем пункте, достаточна для того, чтобы описать возможности дополнительных случаев совмещения в памяти элементов массивов и объектов из общих блоков. Не допускается явное или неявное отведение одной и той же единицы памяти для хранения более чем одного элемента одного и того же массива.

7.2.1.5. **Объявление внешних имен.** Объявление внешних имен имеет вид

$$\text{EXTERNAL } v_1, v_2,\ldots,v_n$$

(каждое v_1— имя внешней процедуры).

Появление некоторого имени в этом объявлении означает, что это имя является именем внешней процедуры. Если имя внешней процедуры используется в некотором программном модуле в качестве фактического параметра, то оно должно появиться в этом модуле в объявлении внешних имен.

7.2.1.3, 7.2.1.3.1, 7.2.1.4, 7.2.1.5. **(Измененная редакция, Изм. № 1).**

7.2.1.6. **Объявление типа.** Объявление типа имеет вид

$$t\ v_1, v_2,\ldots,v_n$$

(t—INTEGER, REAL, DOUBLE PRECISION, COMPLEX или LOGICAL;

каждое v_i— имя переменной, имя массива, имя функции или описание массива).

Объявление типа используется для того, чтобы изменить или подтвердить неявное указание типа, а также для предписания величинам типа двойной точности, комплексный или логический. Это объявление может также содержать информацию о структуре массива.

Появление символического имени v_i в объявлении типа означает, что все вхождения имени v_i в программный модуль связываются с определенным типом данных (см. п. 8.3.1).

7.2.2. **Объявление начальных данных.** Объявление начальных данных имеет вид

$$\text{DATA } k_1/d_1/,\ k_2/d_2/,\ldots,\ k_n/d_n/$$

(каждое k_1 — список, содержащий имена переменных и (или) элементов массивов;

каждое d_1 — список констант, перед каждой из которых может быть записана конструкция j*;

j — *отличное от нуля целое без знака*).

Если какой-либо из списков состоит более чем из одного элемента, то эти элементы отделяются друг от друга запятой.

Формальные параметры не могут входить в списки k. Любое индексное выражение должно быть целым без знака.

Наличие конструкции j* перед константой эквивалентно записи этой константы j раз подряд через запятую. В списках d_1 могут встречаться текстовые константы.

Объявление начальных данных используется для придания начальных значений некоторым переменным и элементам массивов. В каждой паре этих списков должно существовать взаимнооднозначное соответствие между именами списка k_1 и константами списка d_1 — начальные значения устанавливаются на основе этого соответствия. Переменная или элемент массива, которым придаются начальные значения, не могут входить в непомеченный общий блок. Переменной или элементу массива, входящим в помеченный общий блок, начальные значения могут быть приданы только в модуле-блоке данных.

7.2.3. **Объявление формата.** Объявления формата используются в связи с форматным вводом/выводом для указания необходимого преобразования и редактирования информации при переходе от ее внутреннего представления к внешней последовательности символов и обратно.

Объявление формата имеет вид

$$\text{FORMAT } (q_1t_1z_1t_2z_2\ldots z_{n-1}t_nq_2)$$

(каждое q_1 — серия дробных черт или пусто;

каждое t_1 — описатель поля или группа описателей полей;

каждое z_1 — разделитель полей;

n может быть равно нулю;

$(q_1t_1z_1t_2z_2\ldots z_{n-1}t_nq_2)$ — спецификация формата).

Каждое объявление формата должно быть помечено.

7.2.3.1. **Описатели полей.** Описатели полей формата могут иметь вид

srFw.d
srEw.d
srGw.d

srDw.d
rIw
rLw
rAw
nHh$_1$h$_2$...h$_n$
nX

(буквы F, E, G, D, I, L, A, H и X указывают способ преобразования и редактирования при переходе от внутреннего представления записей к внешнему и обратно и называются кодами преобразований;

w и n — отличные от нуля целые без знака, указывающие ширину поля для внешней последовательности символов;

d — целое без знака, указывающее количество цифр в дробной части числа, изображаемого внешней последовательностью символов (за исключением кода преобразования G);

r — либо пусто, либо отличное от нуля целое без знака, называемое счетчиком повторений, который указывает, сколько раз должен быть повторен следующий за ним описатель поля;

s — либо пусто, либо конструкция, обозначающая предписатель масштабного множителя (см. п. 7.2.3.5);

каждое h_i — один из символов, представимых в процессоре).

Ширина поля должна быть обязательно указана в каждом описателе поля. Для описателей вида w.d обязательно должно быть указано d, даже в том случае, когда оно есть нуль; w должно быть больше либо равно d.

Термин «основной описатель поля» используется для обозначения такого описателя поля, в котором отсутствуют s и r.

Внутреннее представление внешних полей соответствует внутреннему представлению констант соответствующих типов (пп. 4.2 и 5.1.1).

7.2.3.2. **Разделители полей.** Разделители полей предназначены для разделения описателей полей и (или) групп описателей полей. Разделителями полей являются дробная черта и запятая. Серия дробных черт также является разделителем полей.

Дробная черта используется не только для разделения описателей полей, но и для разграничения форматных записей. Форматная запись — это последовательность символов. Длины таких последовательностей для данного внешнего носителя зависят как от процессора, так и от внешнего носителя.

Завершение обработки тех символов, которые могут содержаться в записи на внешнем носителе, само по себе не вызывает ввода или начала обработки следующей записи.

7.2.3.3. **Спецификации повторений.** Повторение какого-либо описателя поля (за исключением nH и nX) достигается при помощи счетчика повторений: если позволяет список ввода/вывода, то заданное преобразование выполняется повторно указанное число раз.

Повторение группы описателей полей или разделителей полей достигается заключением их в скобки и, возможно, помещением перед левой скобкой целого числа без знака, называемого счетчиком повторений группы и задающего число повторений взятой в скобки группы. Если счетчик повторений группы не задан, то число повторений принимается равным единице. Рассмотренная форма называется основной группой.

Более сложные группы можно образовать заключением в скобки описателей полей, разделителей полей или основных групп. Для такой группы также может быть задан счетчик повторений. Скобки, в которые заключается вся спецификация формата, не рассматриваются как скобки, ограничивающие группу.

7.2.3.4. **Взаимодействие форматного управления со списком ввода/вывода.** Начало выполнения оператора форматного ввода или форматного вывода инициирует форматное управление. Каждое действие форматного управления определяется как очередным элементом списка ввода/вывода, если таковой имеется, так и очередным описателем поля в спецификации формата. Если имеется список ввода/вывода, то в спецификации формата должен существовать по крайней мере один описатель поля, отличный от nH и nX.

Если под форматным управлением выполняется оператор ввода, то при инициировании форматного управления читается одна запись; после этого следующие записи читаются только в том случае, если этого требует спецификация формата. Каждое такое действие не может требовать большего числа символов, чем содержится в очередной записи.

Если под форматным управлением выполняется оператор вывода, то передача записи происходит каждый раз, когда спецификация формата требует перехода к новой записи. Завершение форматного управления влечет за собой вывод очередной записи.

Спецификация формата (если не считать эффекта применения счетчиков повторений) интерпретируется слева направо.

Каждому основному описателю I, F, E, G, D, A или L, интерпретируемому в спецификации формата, соответствует один элемент, задаваемый списком ввода/вывода, за исключением элемента типа комплексный, который требует интерпретации двух основных описателей F, E или G. Каждому основному описателю H или X не соответствует никакой элемент, задаваемый списком ввода/вывода; в этом случае форматное управление связывает информацию, содержащуюся в таком описателе, непосредственно с записью. Если встречается дробная черта, то спецификация формата требует начала новой и/или окончания очередной записи. Если при выполнении оператора ввода форматное управление завершается или встречается дробная черта, то все необработанные символы очередной записи пропускаются.

Как только форматное управление встречает в спецификации формата основной описатель I, F, E, G, D, A или L, оно проверяет, имеется ли соответствующий элемент, задаваемый списком ввода/вывода. Если такой элемент существует, то форматное управление передает преобразованную соответствующим образом информацию от элемента к записи или наоборот и продолжает работу. Если соответствующего элемента нет, то форматное управление завершается.

Если форматное управление дойдет до последней внешней правой скобки спецификации формата, то проверяется, остались ли необработанные элементы в списке ввода/вывода. Если таких элементов нет, то управление завершается; если же элементы в списке остались, то форматное управление требует начать новую запись и управление возвращается к той спецификации повторений группы, которая заканчивается последней предшествующей правой скобкой, а при ее отсутствии — к первой левой скобке спецификации формата. Заметим, что такое действие само по себе не влияет на значение масштабного множителя (п. 7.2.3.5).

7.2.3, 7.2.3.1—7.2.3.4. **(Измененная редакция, Изм. № 1).**

7.2.3.5. **Масштабный множитель.** Предписатель масштабного множителя определен для использования с преобразованиями F, E, G и D и имеет вид

nP

(n — масштабный множитель — есть целое без знака, перед которым может располагаться знак минус).

При инициировании форматного управления масштабный множитель устанавливается равным нулю. Будучи однажды установленным, масштабный множитель применяется ко всем интерпретируемым впоследствии описателям полей F, E, G и D до тех пор, пока не встретится другой предписатель масштабного множителя — с этого момента будет считаться установленным новый масштабный множитель.

7.2.3.5.1. **Действие масштабного множителя.** Масштабный множитель влияет на соответствующие преобразования следующим образом.

Для преобразований F, E, G и D при вводе (в предположении, что во внешнем поле отсутствует экспонента), а также для преобразования F при выводе, эффект масштабного множителя состоит в том, что число во внешнем представлении равно числу во внутреннем представлении, умноженному на 10^n.

Для F, E, G и D при вводе масштабный множитель не задает никакого действия, если во внешнем поле присутствует экспонента.

Для E и D при выводе часть числа, образующая смешанную дробь (п. 5.1.1.2), умножается на 10^n, а значение экспоненты умножается на 10^{-n}.

Для G при выводе действие масштабного множителя временно прекращается, если значение, подвергаемое преобразованию, находится в области, допускающей использование преобразования F; если требуется использование преобразования E, то масштабный множитель задает те же действия, что и для E при выводе.

7.2.3.6. **Преобразования чисел.** Описатели числовых полей I, F, E, G и D используются для задания ввода/вывода данных типа целый, вещественный, двойной точности и комплексный.

Для всех таких преобразований при вводе ведущие пробелы являются незначащими, а остальные пробелы трактуются как нули. Знак плюс может быть опущен. Поле, состоящее из одних пробелов, трактуется как нуль.

Для преобразований F, E, G и D при вводе точка, присутствующая в поле ввода, отменяет задание точки в описателе поля.

Для всех таких преобразований при выводе поле вывода выравнивается вправо. Если число символов, порождаемых преобразованием, меньше ширины поля, то в поле вывода вставляются ведущие пробелы.

Для всех таких преобразований при выводе внешнее представление отрицательного значения должно иметь знак минус, представление положительного значения может иметь знак плюс.

Число символов, порождаемых преобразованием при выводе, не должно превышать ширины поля.

7.2.3.6.1. **Преобразование данных типа целый.** Описатель числового поля Iw указывает, что внешнее поле — это целое число, занимающее w позиций. Значение элемента списка является или (после ввода) должно являться во внутреннем представлении данным типа целый.

Во внешнем поле ввода последовательность символов должна представлять целое число без знака или со знаком (п. 5.1.1.1), с учетом трактовки пробелов (п. 7.2.3.6).

Внешнее поле вывода состоит, если это необходимо, из пробелов, за которыми следует знак минус, если данное во внутреннем представлении отрицательно, или, возможно, знак плюс в противном случае, за которым следует целое число без знака, представляющее величину внутреннего данного.

7.2.3.5.1, 7.2.3.6, 7.2.3.6.1. **(Измененная редакция, Изм. № 1).**

7.2.3.6.2. **Преобразование данных типа вещественный.** Имеются три преобразования, применимые к данным типа вещественный: F, E, G.

7.2.3.6.2.1. Описатель числового поля Fw.d указывает, что внешнее поле занимает w позиций, а дробная часть состоит из d цифр. Значение элемента списка является или (после ввода) должно являться во внутреннем представлении данным типа вещественный.

Основная форма внешнего поля ввода состоит из последовательности цифр, возможно, содержащей точку, а этой последовательности может предшествовать знак. За основной формой может следовать экспонента в одной из следующих форм:

целое число со знаком;

E, за которой следует целое число без знака или со знаком;

D, за которой следует целое число без знака или со знаком.

Экспонента, содержащая D, эквивалентна экспоненте, содержащей E.

Внешнее поле вывода состоит, если это необходимо, из пробелов, за которыми следует знак минус, если внутреннее значение отрицательно, или, возможно, знак плюс в противном случае, за которым следует последовательность цифр, содержащая точку и представляющая величину внутреннего данного, измененную установленным масштабным множителем и округленную до d десятичных цифр после запятой.

7.2.3.6.2.2. Описатель числового поля Ew.d указывает, что внешнее поле занимает w позиций, а дробная часть состоит из d цифр. Значение элемента списка является или (после ввода) должно являться во внутреннем представлении данным типа вещественный.

Форма внешнего поля ввода такая же, что и для преобразования F.

Стандартная форма внешнего поля вывода для масштабного множителя, равного нулю, имеет вид

$$\xi 0.x_1...x_d Y$$

(ξ — отсутствие символа, либо знак минус в этой позиции;

$x_1...x_d$ — старшие d десятичных цифр округленного выводимого значения данного;

Y — десятичная экспонента;

цифра 0 в приведенной выше стандартной форме может быть опущена).

Y имеет вид

$$E \pm y_1 y_2$$

или

$$\pm y_1 y_2 y_3$$

(y_i — цифра).

Вместо знака плюс в первом из этих видов может стоять пробел.

Масштабный множитель n управляет десятичной нормализацией между смешанной дробью $\xi 0.x_1...x_d$ и экспонентой следующим образом (см. также п. 7.2.3.5.1).

Если $n \leqslant 0$, то будет ровно $-n$ ведущих нулей и $d+n$ значащих цифр после точки.

Если $n>0$, то будет ровно n значащих цифр слева от точки $d-n+1$ цифр справа от точки.

7.2.3.6.2.3. Описатель числового поля G w.d указывает, что внешнее поле занимает w позиций с d значащими цифрами. Значение элемента списка является или (после ввода) должно являться во внутреннем представлении данным типа вещественный.

Метод представления внешней последовательности символов при выводе зависит от величины вещественного данного, подвергаемого преобразованию. Пусть N — величина внутреннего данного. Соответствие между N и эквивалентным применяемым методом преобразования задается следующим образом:

Величина данного	Преобразование
$0.1\leqslant N<1$	F(w—4).d,4X
$1\leqslant N<10$	F(w—4).(d—1),4X
⋮	⋮
$10^{d-2}\leqslant N<10^{d-1}$	F(w—4).1,4X
$10^{d-1}\leqslant N<10^{d}$	F(w—4).0,4X

В остальных случаях sEw.d.

Заметим, что действие масштабного множителя временно приостанавливается, если величина данного, подвергаемого преобразованию, оказывается в области, допускающей использование преобразования F.

7.2.3.6.2.1—7.2.3.6.2.3 **(Измененная редакция, Изм. № 1).**

7.2.3.6.3. **Преобразование данных типа двойной точности.** Описатель числового поля Dw.d указывает, что внешнее поле занимает w позиций, а дробная часть состоит из d цифр. Значение элемента списка является или (после ввода) должно являться во внутреннем представлении данным типа двойной точности.

Основная форма внешнего поля ввода та же самая, что и для преобразования данных типа вещественный.

Внешнее поле вывода такое же, что и для преобразования E, за исключением того, что символ E в экспоненте может заменяться символом D.

7.2.3.6.4. **Преобразование данных типа комплексный.** Поскольку данное типа комплексный состоит из пары отдельных данных типа вещественный, то его преобразование определяется двумя последовательно интерпретируемыми описателями вещественных полей. Первый из них дает вещественную часть, второй — мнимую.

7.2.3.7. **Преобразование данных типа логический.** Описатель логического поля Lw указывает, что внешнее поле занимает w позиций как последовательность символов, вид которой определен ниже. Элемент списка является или (после ввода) должен являться во внутреннем представлении данным типа логический.

Внешнее поле ввода должно состоять из, возможно, пробелов, за которыми следует символ T либо символ F, за которым, возмож-

но, следуют остальные символы, изображающие значения «истина» (True) или «ложь» (False) соответственно.

Внешнее поле вывода состоит из w—1 пробела, за которым следует символ T или символ F, если значение внутреннего данного есть соответственно «истина» или «ложь».

7.2.3.8. **Описатель текстового поля.** Текстовая информация может передаваться посредством двух описателей полей: nH или Aw.

Описатель nH указывает, что в качестве текстовой информации, подлежащей передаче при вводе или выводе, берутся n символов (включая пробелы), следующие непосредственно за описателем поля nH в самой спецификации формата.

Описатель Aw указывает, что при вводе прочитываются w символов и передаются в элемент, определяемый списком ввода/вывода, а при выводе из этого элемента выбираются w символов.

Пусть g — число символов, которые могут быть размещены в одной единице памяти. Если ширина поля w больше либо равна g, то при вводе из внешнего поля будут взяты самые правые g символов. Если ширина поля w меньше g, то при вводе во внутреннем представлении будет w символов, сдвинутых влево, за которыми будет следовать g—w пробелов.

Если ширина поля w больше g, то при выводе внешнее поле будет состоять из w—g пробелов, за которыми следуют g символов из внутреннего представления. Если ширина поля w меньше либо равна g, то при выводе внешнее поле будет состоять из w самых левых символов из внутреннего представления.

7.2.3.9. **Описатель поля пробелов.** Описатель поля пробелов есть nX. При вводе пропускаются n символов из внешней вводимой записи, а при выводе во внешнюю выводимую запись вставляется n пробелов.

7.2.3.7—7.2.3.9. **(Измененная редакция, Изм. № 1).**

7.2.3.10. **Спецификация формата в массивах.** Любой оператор форматного ввода/вывода может содержать имя массива в том месте, которое отведено для указания метки объявления формата. В тот момент, когда на этот массив производится ссылка из оператора ввода/вывода, начальная часть информации, содержащейся в этом массиве, взятая в естественном порядке, должна образовывать правильную спецификацию формата. На информацию, содержащуюся в массиве и следующую за правой скобкой, ограничивающей спецификацию формата, не накладывается никаких ограничений.

Спецификация формата, размещаемая в массиве, имеет тот же вид, который был определен для объявления формата, т. е. она начинается с левой скобки и заканчивается правой скобкой. В массиве описатель поля nH не может быть частью спецификации формата. Спецификация формата может быть помещена в массив с помощью объявления начальных данных, а также с по-

мощью оператора ввода с форматом, содержащим описатель поля типа А.

8. ПРОЦЕДУРЫ И МОДУЛИ

Имеются четыре категории процедур: внутренние функции, встроенные функции, внешние функции и внешние подпрограммы. Первые три категории процедур относятся к функциям (или процедурам-функциям), а последняя категория — к подпрограммам (или процедурам-подпрограммам). Имеются две категории модулей: модули-процедуры и модули-спецификации; к первой из них относятся модули-функции и модули-подпрограммы, а ко второй — модули-блоки данных. Правила типа для процедур-функций указаны в п. 5.3.

8.1. **Внутренние функции.** Внутренняя функция определяется в том же самом программном модуле, в котором имеются ссылки на эту функцию. Такая функция определяется при помощи объявления внутренней функции.

В каждом программном модуле все определения внутренних функций должны предшествовать первому оператору этого модуля и следовать за объявлениями спецификаций, если таковые имеются. Имя внутренней функции не должно встречаться в том же самом программном модуле ни в объявлениях внешних имен, ни в качестве имени переменной или имени массива.

8.1.1. **Структура объявлений внутренних функций.** Внутренняя функция определяется при помощи объявления внутренней функции, которое имеет вид

$$f(a_1, a_2,...,a_n) = e$$

(f — символическое имя определяемой функции;

e — выражение;

каждое a_1 — символическое имя, называемое формальным параметром).

Соответствие между f и e должно удовлетворять правилам присваивания из пп. 7.1.1.1—7.1.1.2.

Символические имена, являющиеся формальными параметрами, используются в объявлении внутренней функции лишь для указания типа, числа и порядка параметров функции и могут совпадать с именами переменных того же типа, встречающимися где-либо еще в рассматриваемом программном модуле (но вне объявления данной функции). Все формальные параметры в одном объявлении функции должны быть различными.

Выражение e, кроме формальных параметров, может содержать только:

нетекстовые константы;

указатели переменных;

указатели встроенных функций;

указатели внутренних функций, определенных ранее в данном программном модуле;

указатели внешних функций.

8.1.2. **Ссылки на внутренние функции.** Для ссылки на внутреннюю функцию используется ее указатель (п. 5.2.) в качестве первичного выражения в арифметическом или логическом выражении. Фактические параметры, образующие список фактических параметров в указателе функции, должны согласовываться по порядку, числу и типу с соответствующими формальными парамет-

Таблица 3

Встроенная функция	Определение	Число параметров	Символическое имя	Тип	
				параметров	функции
Абсолютное значение	$\|a\|$	1	ABS	Вещ.	Вещ.
			IABS	Цел.	Цел.
			DABS	Дв.	Дв.
Усечение*	sign (a) умножается на [$\|a\|$]	1	AINT	Вещ.	Вещ.
			INT	Вещ.	Цел.
			IDINT	Дв.	Цел.
Взятие остатка**	$a_1 (\mathrm{mod}\ a_2)$	2	AMOD	Вещ.	Вещ.
			MOD	Цел.	Цел.
Выбор наибольшего значения	max (a_1, a_2...)	$\geqslant 2$	AMAX0	Цел.	Вещ.
			AMAX1	Вещ.	Вещ.
			MAX0	Цел.	Цел.
			MAX1	Вещ.	Цел.
			DMAX1	Дв.	Дв.
Выбор наименьшего значения	min (a_1, a_2...)	$\geqslant 2$	AMIN0	Цел.	Вещ.
			AMIN1	Вещ.	Вещ.
			MIN0	Цел.	Цел.
			MIN1	Вещ.	Цел.
			DMIN1	Дв.	Дв.
Преобразование в плавающую форму	Преобразование от целого к вещественному	1	FLOAT	Цел.	Вещ.
Преобразование в фиксированную форму	Преобразование от вещественного к целому	1	IFIX	Вещ.	Цел.

Продолжение табл. 3

Встроенная функция	Определение	Число параметров	Символическое имя	Тип параметров	Тип функции
Передача знака	sign (a_2) умножается на $\lvert a_1 \rvert$	2	SIGN	Вещ.	Вещ.
			ISIGN	Цел.	Цел.
			DSIGN	Дв.	Дв.
Положительная разность	$a_1 - \min(a_1, a_2)$	2	DIM IDIM	Вещ. Цел.	Вещ. Цел.
Получение максимальной значащей части аргумента двойной точности		1	SNGL	Дв.	Вещ.
Получение вещественной части комплексного аргумента		1	REAL	Компл.	Вещ.
Получение мнимой части комплексного аргумента		1	AIMAG	Компл.	Вещ.
Преобразование вещественного аргумента в форму двойной точности		1	DBLE	Вещ.	Дв.
Преобразование двух вещественных аргументов в комплексную форму	$a_1 + a_2\sqrt{-1}$	2	CMPLX	Вещ.	Компл.
Получение комплексной величины, сопряженной с аргументом		1	CONJG	Компл.	Компл.

* $[x]$ означает наибольшее целое, величина которого не превосходит величины x, имеющее тот же знак, что и x.

** Функция MOD или AMOD (a_1, a_2) определяется как

$$a_1 - [a_1 / a_2] \cdot a_2.$$

рами. Фактическим параметром в указателе внутренней функции может быть любое выражение того же типа, что и соответствующий формальный параметр.

Вычисление значения указателя внутренней функции заключается в установлении связи (п. 10.2.2) между значениями фактических параметров и соответствующими формальными параметрами и в вычислении значения выражения в объявлении, определяющем эту функцию. Вычисленное значение выражения принимается в качестве значения указателя функции и тем самым становится доступным в выражении, содержащем этот указатель.

8.2. **Встроенные функции и ссылки на них.** Символические имена встроенных функций заранее известны процессору и, если они удовлетворяют условиям, изложенным в п. 10.1.7, им приписывается специальный смысл и тип в соответствии с табл. 3.

Для ссылки на встроенную функцию используется ее указатель в качестве первичного выражения в арифметическом или логическом выражении. Фактические параметры, образующие список фактических параметров в указателе функции, должны согласовываться по порядку, числу и типу со спецификациями, данными в табл. 3, и могут быть любыми выражениями соответствующих типов. Значение встроенных функций AMOD, MOD, SIGN, ISIGN и DSIGN не определено, если значение второго параметра равно нулю.

Вычисление значения указателя встроенной функции заключается в выполнении действий, определенных в табл. 3, над значениями фактических параметров. Результирующее значение принимается в качестве значения указателя функции и тем самым становится доступным в выражении, содержащем этот указатель.

(Измененная редакция, Изм. № 1).

8.3. **Внешние функции.** Внешняя функция определяется вне того программного модуля, который ссылается на нее. Для определения внешней функции средствами ФОРТРАНа служит отдельный программный модуль, называемый модулем-функцией, первым предложением этого модуля является заголовок функции.

8.3.1. **Структура модулей-функций.** Заголовок функции имеет вид

$$t\ \text{FUNCTION}\ f(a_1,a_2,\ldots a_n)$$

(t — либо пусто, либо INTEGER, либо REAL, либо DOUBLE PRECISION, либо COMPLEX, либо LOGICAL;

f — символическое имя определяемой функции;

каждое a_i — символическое имя, называемое формальным параметром).

Каждый формальный параметр должен быть либо именем переменной, либо именем массива, либо именем внешней процедуры. Все формальные параметры в одном заголовке функции должны быть различными.

Модули-функции строятся в соответствии с п. 9.1.6 со следующими ограничениями. Символическое имя определяемой функции должно также встречаться в этом модуле в качестве имени переменной. При каждом выполнении модуля эта переменная должна быть определена, причем впоследствии на эту переменную можно ссылаться или переопределять ее (см. разд. 10). Значение этой переменной к моменту выполнения в данном модуле любого оператора возврата называется значением определяемой функции. Символическое имя определяемой функции не должно встречаться ни в одном объявлении этого программного модуля, за исключением вхождения в заголовок функции в качестве ее имени. Формальные параметры не могут встречаться в объявлениях эквивалентности, общих объектов или начальных данных в этом модуле-функции. Модуль-функция при своем выполнении может определять или переопределять один или несколько своих фактических параметров, что, наряду с вычислением значения функции, также является результатом выполнения модуля-функции. Модуль-функция может содержать любые предложения, за исключением заголовка блока данных, заголовка подпрограммы, другого заголовка функции, а также любого предложения, которое явно или неявно ссылается на определяемую функцию. Модуль-функция должен содержать по крайней мере один оператор возврата.

8.3.2. **Ссылки на внешние функции.** Для ссылки на внешнюю функцию используется ее указатель (п. 5.2) в качестве первичного выражения в арифметическом или логическом выражении. Фактические параметры, образующие список фактических параметров в указателе функции, должны согласовываться по порядку, числу и типу с соответствующими формальными параметрами модуля-функции. Фактическим параметром в указателе внешней функции может быть:

имя переменной;

имя элемента массива;

имя массива;

любое другое выражение;

имя внешней процедуры.

Если фактический параметр является именем внешней функции или именем подпрограммы, то соответствующий формальный параметр должен употребляться как имя внешней функции или как имя подпрограммы.

Если фактический параметр соответствует формальному параметру, который определяется или переопределяется в данной подпрограмме, то фактический параметр должен быть либо именем

переменной, либо именем элемента массива, либо именем массива. Вычисление значения указателя внешней функции производится путем обращения к соответствующему модулю-функции. При этом обращении устанавливается связь (п. 10.2.2) между фактическими параметрами и всеми вхождениями формальных параметров в операторы, в объявления внутренних функций, а также в качестве регулируемых размеров в том модуле, который определяет указываемую функцию. Если фактический параметр является выражением, то эта связь параметров устанавливается по значению, а не по наименованию. После установления этих связей производится выполнение модуля, начиная с первого по порядку его оператора.

Если фактический параметр является именем элемента массива, содержащим переменные в индексе, то при установлении связи с формальным параметром этот фактический параметр может быть заменен на такой же параметр, но с постоянным индексом, содержащим те значения, которые были бы получены при вычислении переменного индекса непосредственно перед установлением связи между параметрами.

Если формальный параметр внешней функции используется в качестве имени массива, то соответствующий фактический параметр должен быть именем массива или именем элемента массива (п. 10.1.3).

Если ссылка на функцию приводит к тому, что ее формальный параметр становится связанным (п. 10.2.2) с другим формальным параметром этой же функции или с объектом общего блока, то в этом модуле-функции запрещено определение всех связанных таким способом объектов.

8.3.3. **Основные внешние функции.** Процессоры ФОРТРАНа должны быть снабжены внешними функциями, перечисленными в табл. 4. Ссылки на эти функции производятся так же, как было описано в п. 8.3.2. Фактические параметры, для которых значение этих функций математически не определено или имеет тип, отличный от приведенного в табл. 4, являются недопустимыми.

Таблица 4

Основная внешняя функция	Определение	Число параметров	Символическое имя	Тип	
				параметров	функции
Экспоненциальная функция	e^a	1	EXP DEXP CEXP	Вещ. Дв. Компл.	Вещ. Дв. Компл.
Натуральный логарифм	ln(a)	1	ALOG DLOG CLOG	Вещ. Дв. Компл.	Вещ. Дв. Компл.
Десятичный логарифм	lg(a)	1	ALOG10 DLOG10	Вещ. Дв.	Вещ. Дв.

Продолжение табл. 4

Основная внешняя функция	Определение	Число параметров	Символическое имя	Тип	
				параметров	функции
Тригонометрический синус	sin(a)	1	SIN DSIN CSIN	Вещ. Дв. Компл.	Вещ. Дв. Компл.
Тригонометрический косинус	cos(a)	1	COS DCOS CCOS	Вещ. Дв. Компл.	Вещ. Дв. Компл.
Гиперболический тангенс	tanh(a)	1	TANH	Вещ.	Вещ.
Квадратный корень	$a^{1/2}$	1	SQRT DSQRT CSQRT	Вещ. Дв. Компл.	Вещ. Дв. Компл.
Арктангенс	arctan(a) arctan(a_1/a_2)	1 2	ATA DATAN ATAN2 DATAN2	Вещ. Дв. Вещ. Дв.	Вещ. Дв. Вещ. Дв.
*Взятие остатка**	$a_1 (mod\ a_2)$	2	DMOD	Дв.	Дв.
Модуль		1	CABS	Компл.	Вещ.

* Функция DMOD(a_1, a_2) определяется как $a_1 - [a_1/a_2] \cdot a_2$, где [x] означает наибольшее целое, величина которого не превосходит величины x, имеющее тот же знак, что и x.

8.3.2, 8.3.3. **(Измененная редакция, Изм. № 1).**

8.4. **Подпрограммы.** Внешняя подпрограмма определяется вне того программного модуля, который ссылается на нее. Для определения внешней подпрограммы средствами ФОРТРАНа служит отдельный программный модуль, называемый модулем-подпрограммой; первым предложением этого модуля является заголовок *подпрограммы.*

8.4.1. **Структура модулей-подпрограмм.** Заголовок подпрограммы имеет вид

SUBROUTINE s(a_1, a_2,...,a_n)

или

SUBROUTINE s

(s — символическое имя определяемой подпрограммы;
каждое a_i — символическое имя, называемое формальным параметром).

Каждый формальный параметр должен быть либо именем переменной, либо именем массива, либо именем внешней процедуры. Все формальные параметры в одном заголовке подпрограммы должны быть различными.

Модули-подпрограммы строятся в соответствии с п. 9.1.6 со следующими ограничениями. Символическое имя определяемой подпрограммы не должно встречаться ни в одном предложении определяющего ее программного модуля, за исключением вхождения этого имени в заголовок подпрограммы в качестве ее имени. Внутри этого модуля формальные параметры не могут встречаться в объявлениях эквивалентности, общих объектов и начальных данных. При своем выполнении модуль-подпрограмма может определять или переопределять свои фактические параметры, что также является результатом выполнения модуля-подпрограммы. Модуль-подпрограмма может содержать любые предложения, за исключением заголовка блока данных, заголовка функции и другого заголовка подпрограммы, а также любого предложения, которое явно или неявно ссылается на определяемую подпрограмму. Модуль-подпрограмма должен содержать по крайней мере один оператор возврата.

8.4.2. **Ссылки на внешние подпрограммы.** Для ссылки на внешнюю подпрограмму используется оператор вызова подпрограммы (п. 7.1.2.4). Фактические параметры, образующие список фактических параметров этого оператора, должны согласовываться по порядку, числу и типу с соответствующими формальными параметрами модуля-подпрограммы. Исключением из правила согласования типов является использование в качестве фактического параметра текстовой константы.

Фактическим параметром при ссылке на внешнюю подпрограмму может быть:

а) текстовая константа;
б) имя переменной;
в) имя элемента массива;
г) имя массива;
д) любое другое выражение;
е) имя внешней процедуры.

Если фактический параметр является именем внешней функции или именем подпрограммы, то соответствующий формальный параметр должен также использоваться в качестве имени внешней функции или имени подпрограммы соответственно.

Если фактический параметр соответствует формальному параметру, который в подпрограмме определяется или переопреде-

ляется, то фактический параметр должен быть либо именем переменной, либо именем элемента массива, либо именем массива.

Выполнение оператора вызова подпрограммы производится путем обращения к соответствующему модулю-подпрограмме. При этом обращении устанавливается связь (п. 10.2.2) между фактическими параметрами и всеми вхождениями формальных параметров в операторы, объявления внутренних функций, а также в качестве регулируемых размеров в модуле-подпрограмме. Если фактический параметр относится к категории, приведенной в подпункте *д* в указанном выше перечне допустимых фактических параметров, то эта связь параметров устанавливается по значению, а не по наименованию. После установления связи параметров производится выполнение модуля, начиная с первого по порядку его оператора.

Если фактический параметр является именем элемента массива, содержащим переменные в индексе, то при установлении связи с формальным параметром этот фактический параметр может быть заменен на такой же параметр, но с постоянным индексом, содержащим те значения, которые были бы получены при вычислении переменного индекса непосредственно перед установлением связи между параметрами.

Если формальный параметр является именем массива, то соответствующий фактический параметр должен быть именем массива или именем элемента массива (п. 10.1.3).

Если вызов подпрограммы приводит к тому, что ее формальный параметр оказывается связанным (п. 10.2.2) с другим формальным параметром этой же подпрограммы или с объектом общего блока, то внутри подпрограммы запрещено определение всех связанных таким образом объектов.

8.5. **Модуль-блок данных.** Заголовок спецификации блока данных имеет вид

BLOCK DATA

Этот заголовок может встречаться только в качестве первого предложения модуля-спецификации, который называется модулем-блоком данных и служит для придания начальных значений элементам помеченных общих блоков. Этот специальный модуль содержит только объявления типа, эквивалентности, массивов, начальных данных и общих объектов.

Если какому-либо элементу данного общего блока было придано начальное значение в таком модуле, то в него должен быть включен полный набор объявлений спецификаций для всего этого общего блока — даже в том случае, если некоторые из элементов блока не присутствуют в объявлениях начальных данных. В одном таком модуле начальные значения могут быть приданы элементам как одного, так и нескольких общих блоков.

8.4.2, 8.5. **(Измененная редакция, Изм. № 1).**

9. ПРОГРАММА

9.1. **Строение программных модулей.** Программный модуль состоит из заголовка модуля и тела модуля.

(Измененная редакция, Изм. № 1).

9.1.1. **Раздел операторов.** Раздел операторов должен содержать хотя бы один оператор. В самом его начале могут (но не обязаны) находиться в произвольном порядке и количестве объявления внутренних функций, начальных данных и (или) форматов. Затем должна следовать непустая совокупность операторов, среди которых могут (но не обязаны) содержаться объявления форматов и (или) начальных данных.

9.1.2. **Тело модуля.** Тело модуля состоит из (возможно, пустой) совокупности объявлений спецификаций и (или) объявлений форматов, за которой следует раздел операторов с заключительной строкой после него.

Таким образом, все объявления спецификаций должны быть расположены до объявлений внутренних функций, а эти последние — до первого оператора. Объявления форматов могут располагаться в теле модуля где угодно.

(Измененная редакция. Изм. № 1).

9.1.3. **Программный модуль.** Программный модуль является либо головным модулем (п. 9.1.4), либо модулем-блоком данных (п. 9.1.5), либо модулем-процедурой (п. 9.1.6).

9.1.4. **Головной модуль.** Головной модуль состоит только из тела модуля.

9.1.5. **Модуль-блок данных.** Модуль-блок данных представляет собой последовательность, образованную заголовком спецификации блока данных, набором соответствующих (п. 8.5) объявлений начальных данных и заключительной строкой, расположенными в указанном порядке.

9.1.6. **Модуль-процедура.** Модуль-процедура состоит из заголовка функции или заголовка подпрограммы, за которым следует тело модуля.

9.1.7. **Выполнимая программа.** Выполнимая программа включает в себя головной модуль, а также произвольное количество других программных модулей и (или) внешних процедур (разд. 2).

(Измененная редакция, Изм. № 1).

9.2. **Нормальный порядок выполнения.** Выполнение любой выполнимой программы начинается с первого по порядку оператора головного модуля. При обращении к программному модулю его выполнение начинается с первого по порядку оператора этого модуля. Если очередной оператор не является оператором перехода, условным арифметическим оператором, оператором возврата, оператором останова или закрывающим оператором тела цикла, то по завершении выполнения этого оператора начинается выполне-

ние непосредственно следующего за ним оператора. Порядок выполнения, определяемый каждым из перечисленных выше операторов, описан в разд. 7. В разделе операторов не может содержаться оператор, который никогда не может быть выполнен.

10. ВНУТРИ- И МЕЖМОДУЛЬНЫЕ СООТНОШЕНИЯ

10.1. **Символические имена.** Символическое имя было определено (п. 3.5) как последовательность, содержащая от одной до шести буквенно-цифровых символов, первый из которых должен быть буквой. Последовательности символов, образующие описатели полей в объявлениях формата или начальные ключевые слова конкретных типов предложений, например GOTO, READ, FORMAT, не являются символическими именами в этих позициях и не образуют начала символических имен. В любом программном модуле символическое имя (возможно, дополненное индексом) должно обозначать элемент одного (и, как правило, только одного) из следующих классов:

I — массив и элементы этого массива;

II — переменная;

III — внутренняя функция;

IV — встроенная функция;

V — внешняя функция;

VI — подпрограмма;

VII — внешняя процедура, которая на основании информации, содержащейся в данном программном модуле, не может быть отнесена ни к подпрограммам, ни к внешним функциям;

VIII — общий блок.

(Измененная редакция, Изм. № 1).

10.1.1. **Соотношения между классами.** Символическое имя из класса VIII некоторого программного модуля может принадлежать также к одному из классов I, II или III того же программного модуля.

Если в программном модуле символическое имя из класса V встречается непосредственно после слова FUNCTION в заголовке функции, то оно обязано принадлежать также классу II того же программного модуля.

Символическое имя, будучи использовано однажды в любом программном модуле выполнимой программы в классе V, VI, VII или VIII, не может употребляться ни в каком другом программном модуле этой выполнимой программы для обозначения другого объекта из этих классов. В совокупности программных модулей, образующей выполнимую программу, имя из класса VII должно быть связано с именем из класса V или VI. Класс VII имеет смысл только по отношению к отдельным программным модулям.

В любом программном модуле никакое символическое имя не может принадлежать более чем одному классу, за исключением

перечисленных выше случаев. На использование символических имен в разных модулях не накладывается никаких дополнительных ограничений.

10.1.2. **Эффект употребления символических имен в объявлениях спецификаций и объявлениях начальных данных.** Символическое имя принадлежит классу I тогда и только тогда, когда оно употребляется в описании массива в качестве имени описания. Для каждого символического имени в одном модуле допускается только одно такое употребление.

Символическое имя, употребленное в объявлении общих объектов (в позиции, отличной от имени блока), принадлежат либо классу I, либо классу II, но не классу V (п. 8.3.1). Для каждого символического имени в одном модуле допускается только одно такое употребление.

Символическое имя, употребленное в объявлении эквивалентности, принадлежит либо классу I, либо классу II, но не классу V (п. 8.3.1).

Символическое имя, употребленное в объявлении типа, не может принадлежать классу VI или VII. Для каждого символического имени в одном модуле допускается только одно такое употребление.

Символическое имя, встречающееся в объявлении внешних имен, принадлежит либо классу V, либо классу VI, либо классу VII. Для каждого символического имени в одном модуле допускается только одно такое употребление.

Символическое имя, встречающееся в объявлении начальных данных, принадлежит либо классу I, либо классу II, но не классу V (п. 8.3.1). Любой единице памяти (п. 7.2.1.3.1) в выполнимой программе начальное значение может придаваться не более одного раза.

10.1.3. **Массив и элемент массива.** За каждым вхождением символического имени, обозначающего какой-либо массив, должен непосредственно располагаться индекс; исключение составляют следующие вхождения:

в список оператора ввода/вывода;

в список формальных параметров;

в список фактических параметров при ссылке на внешнюю процедуру;

в объявление общих объектов;

в объявление типа;

в оператор ввода/вывода в качестве указателя формата.

Формальный параметр внешней процедуры может быть именем массива только в том случае, когда соответствующий фактический параметр при ссылке на эту внешнюю процедуру является именем массива или именем элемента массива. Если фактический параметр является именем массива, то длина массива-формального параметра должна быть не больше длины массива-фактического параметра. Если фактический параметр является именем элемента масси-

ва, то длина массива-формального параметра должна быть не больше увеличенной на единицу разности длины массива-фактического параметра и значения индекса этого элемента массива.

10.1.4. **Внешние процедуры.** Символическое имя принадлежит классу VII, если в некотором программном модуле оно встречается только в объявлении внешних имен и в качестве фактического параметра внешней процедуры.

Формальный параметр некоторой внешней процедуры может быть именем другой внешней процедуры только в том случае, когда соответствующий фактический параметр используется при ссылке на эту другую внешнюю процедуру в качестве ее имени.

При выполнении любой выполнимой программы не допускаются два таких обращения к одному и тому же модулю-процедуре, между которыми не был бы выполнен оператор возврата из этого модуля.

10.1.5. **Подпрограмма.** Символическое имя принадлежит классу VI, если оно встречается непосредственно после слова SUBROUTINE в заголовке подпрограммы или непосредственно после слова CALL в операторе вызова подпрограммы.

10.1.6. **Внутренняя функция.** Символическое имя принадлежит классу III в некотором программном модуле, если оно:

не принадлежит классу I и не встречается в объявлении внешних имен;

за каждым его вхождением, кроме вхождения в объявление типа, непосредственно следует левая скобка;

в рассматриваемом модуле имеется объявление внутренней функции (п. 8.1.1) с этим именем.

10.1.7. **Встроенная функция.** Символическое имя принадлежит классу IV в некотором программном модуле, если оно:

не принадлежит ни классу I, ни классу III и не встречается в объявлении внешних имен;

встречается в колонке имен табл. 3;

не встречается в объявлении типа, который отличается от встроенного типа, определяемого этой таблицей;

за каждым его вхождением (кроме разрешенного выше вхождения в объявление типа) непосредственно следует левая скобка.

Употребление встроенной функции в некотором программном модуле не запрещает использования этого же символического имени для обозначения какого-либо другого объекта в другом программном модуле той же выполнимой программы.

10.1.8. **Внешняя функция.** Символическое имя принадлежит классу V, если оно:

встречается непосредственно после слова FUNCTION в заголовке функции;

не принадлежит ни одному из классов I, III, IV или VI и имеется хотя бы одно его вхождение, непосредственно за которым

следует левая скобка. Левая скобка должна непосредственно следовать за каждым его вхождением, кроме вхождений в объявления типа и внешних имен или в качестве фактического параметра.

(Измененная редакция, Изм. № 1).

10.1.9. **Переменная.** В программном модуле символическое имя принадлежит классу II, если оно:

не принадлежит ни классу VI, ни классу VII, ни классу VIII;

за ним никогда непосредственно не следует левая скобка, если только это имя не следует непосредственно за словом FUNCTION в заголовке функции.

10.1.10. **Имя блока.** Символическое имя принадлежит классу VIII, если оно используется в качестве имени блока в объявлении общих блоков.

10.2. **Понятие определения.** Для числовых величин существует два уровня определения — первый и второй. Понятие определения на первом уровне применимо к любым элементам массивов и переменным; понятие определения на втором уровне — только к переменным типа целый. Для объяснения смысла этих понятий необходимо рассмотреть выполнимую программу в процессе ее выполнения.

Необходимо отметить, что существуют два других способа определения. Первый из них осуществляется оператором предписания для переменной типа целый, но определяет значение типа отличного от целого — он рассмотрен в пп. 7.1.1.3 и 7.1.2.1.2; второй способ, позволяющий ссылаться на внешнюю процедуру, будет рассмотрен в п. 10.2.1.

В дальнейшем, говоря об определении и неопределении применительно к переменным и элементам массивов, мы всегда будем подразумевать первый уровень определения.

10.2.1. **Определение процедур.** Если выполнимая программа содержит информацию, описывающую внешнюю процедуру, то такая внешняя процедура с соответствующим символическим именем определена для использования в этой выполнимой программе. Ссылка на внешнюю функцию или ссылка на подпрограмму (выбирается подходящий случай) с этим символическим именем может встречаться в выполнимой программе, если число параметров в определении и ссылке совпадает. Кроме того, тип внешней функции в определении и ссылке также должен совпадать. Остальные ограничения содержатся в пп. 8.3.1, 8.3.2, 8.4.1, 8.4.2, 10.1.3 и 10.1.4.

Основные внешние функции, перечисленные в п. 8.3.3, определены всегда и на них можно ссылаться при условии выполнения ограничений, указанных выше.

Для указанного использования определено символическое имя из класса III или IV.

10.2.2. **Связи, обеспечивающие определение.** Объекты могут стать связанными посредством:

связи через общие блоки;

связи через эквивалентность;

подстановки параметров.

В результате упомянутых комбинаций может возникнуть множественная связь с одним или несколькими объектами. Любое определение или неопределение одного из элементов множества связанных элементов вызывает определение или неопределение всех элементов этого множества.

С точки зрения определения в любом программном модуле между двумя объектами, каждый из которых встречается в объявлении общих объектов, не существует никакой связи. Кроме того, для общих и эквивалентных объектов не существует никакой связи, кроме установленной в пп. 7.2.1.3.1 и 7.2.1.4.

Если фактический параметр при ссылке на внешнюю процедуру является именем массива, именем элемента массива, или именем переменной, то в соответствии с пп. 10.1.3 и 10.2.1 связь формальных параметров с фактическими определена лишь на время от начала выполнения первого оператора этой процедуры и до начала выполнения первого встреченного оператора возврата этой процедуры. Заметим особо, что эта связь может передаваться более чем через один уровень ссылок на внешние процедуры.

В дальнейшем переменные и элементы массивов, связанные на основании пп. 7.2.1.3.1 и 7.2.1.4, будут считаться эквивалентными только при условии совпадения типов.

Если объект данного типа становится определенным, то в этот же момент все связанные с ним объекты иного типа становятся неопределенными, а все связанные с ним объекты того же самого типа становятся, если не оговорено противное, определенными.

Связь подстановкой параметров допускается лишь при совпадении типов, причем правило таково: объект, порожденный подстановкой параметров, определен в момент входа (в процедуру) только в том случае, если был определен фактический параметр. Если объект, порожденный подстановкой параметров, при выполнении программного модуля становится (пока связь существует) определенным или неопределенным, то соответствующие фактические объекты во всех вызывающих программных модулях становятся соответственно определенными или неопределенными.

10.2.3. **События, обеспечивающие определение.** Переменные и элементы массивов становятся начально определенными в случае, когда их имена связаны при помощи объявления начальных данных с константой того же типа, что и тип соответствующей переменной или массива. Любой объект, не определенный начально, является неопределенным к началу первого выполнения пер-

вого оператора головного модуля. Переопределение определенного объекта допускается всегда, за исключением некоторых случаев для переменных типа целый (пп. 7.1.2.8, 7.1.3.2.1 и 7.2.1.1.2), а также объектов программных модулей (пп. 6.4, 8.3.2 и 8.4.2).

10.2.3.1. Переменные и элементы массивов становятся определенными (или переопределенными) в следующих случаях:

завершение выполнения арифметического или логического оператора присваивания вызывает определение объекта, расположенного слева от знака равенства;

при выполнении оператора ввода каждый объект, которому присваивается значение соответствующего типа из внешней среды ввода, становится определенным в момент этого присваивания. Связанные объекты одинакового типа становятся определенными лишь по завершении выполнения этого оператора ввода;

завершение выполнения оператора цикла способом, приведенным в п. 7.1.2.8.2*е*, вызывает определение управляющей переменной этого цикла;

начало выполнения действий, определяемых списком с циклом, вызывает определение управляющей переменной этого списка.

10.2.3.2. Переменные и элементы массивов становятся неопределенными в следующих случаях:

в момент завершения оператора цикла способом, приведенным в п. 7.1.2.8.2*г*, управляющая переменная этого цикла становится неопределенной;

завершение выполнения оператора предписания вызывает неопределенность переменной, входящей в этот оператор;

при выполнении модулей-функций некоторые объекты (п. 10.2.9) могут стать неопределенными;

завершение выполнения действий, определяемых списком с циклом, вызывает неопределенность управляющей переменной этого списка;

когда объект, связанный с данным объектом и имеющий иной тип, становится определенным;

когда объект, связанный с данным объектом и имеющий такой же тип, становится неопределенным.

10.2.4. **Объекты непомеченного общего блока.** Объекты непомеченного общего блока и связанные с ним объекты не могут быть начально определены.

Такие объекты, ставшие однажды определенными по любому из приведенных выше правил, остаются определенными до тех пор, пока не станут неопределенными в соответствии с этими же правилами.

10.2.5. **Объекты помеченного общего блока.** Объекты помеченного общего блока и связанные с ним объекты могут быть начально определены.

Говорят, что программный модуль содержит имя помеченного общего блока, если это имя встречается в этом программном модуле в качестве имени блока. Если программный модуль содержит имя помеченного общего блока, то любой объект этого блока (и связанные с ним объекты), ставшие однажды определенными по любому из приведенных в п. 10.2.3 правил, остаются определенными до тех пор, пока не станут неопределенными в соответствии с этими же правилами.

Необходимо заметить, что переопределение начально определенного объекта не означает, что этот объект не может впоследствии стать неопределенным. Особо отметим, что если в некотором программном модуле содержится имя помеченного общего блока, не содержащееся ни в одном программном модуле, прямо или косвенно ссылающемся на этот модуль, то выполнение оператора возврата этого модуля вызывает неопределенность всех объектов данного блока и связанных с ними объектов, за исключением начально определенных объектов, сохраняющих свои начально определенные значения.

10.2.6. **Объекты не из общего блока.** Любой объект не из общего блока, кроме формального параметра или значения функции, может быть начально определен.

Такие объекты, ставшие однажды определенными по любому из приведенных выше правил, остаются определенными до тех пор, пока не станут неопределенными в соответствии с этими же правилами.

Если такой объект используется в программном модуле, то завершение выполнения оператора возврата этого модуля вызывает неопределенность всех таких объектов и связанных с ними в этот момент объектов (за исключением начально определенных, которые не были переопределены и не стали неопределенными). В связи с этим необходимо подчеркнуть, что связь между формальными и фактическими параметрами прекращается в момент начала выполнения оператора возврата.

Необходимо подчеркнуть, что переопределение начально определенного объекта не означает, что этот объект впоследствии не может стать неопределенным.

10.2.7. **Основа.** Основой в программном модуле называется последовательность, состоящая из одного или более операторов, которая определяется следующим образом.

10.2.7.1. Конечными операторами основы являются:

оператор цикла;

оператор вызова подпрограммы;

оператор перехода любого типа;

условный арифметический оператор;

оператор останова;

оператор возврата;

первый из операторов (если он существует), непосредственно предшествующих оператору, метка которого употребляется в операторе перехода или условном арифметическом операторе;

арифметический оператор присваивания, в котором слева от знака равенства стоит переменная типа целый;

оператор ввода, в списке которого содержится переменная типа целый;

условный логический оператор, содержащий любой из перечисленных выше допустимых операторов.

10.2.7.2. Начальными операторами основы являются:

первый оператор в программном модуле;

первый из операторов (если он существует), непосредственно следующий за конечным оператором некоторой другой основы.

Каждый начальный оператор основы задает некоторую основу. Если этот начальный оператор является также и конечным оператором основы, то она состоит лишь из одного этого оператора. Если же это не так, то основа состоит из начального оператора и всех последующих операторов до конечного оператора основы включительно.

10.2.7.3. **Последний оператор.** В любом программном модуле последний по порядку оператор должен быть либо оператором перехода, либо условным арифметическим оператором, либо оператором останова, либо оператором возврата. Последний оператор не может быть частью условного логического оператора.

10.2.8. **Второй уровень определения.** В момент использования в индексах и вычисляемых операторах перехода переменные типа целый должны быть определены на втором уровне.

10.2.8.1. Если при выполнении некоторой основы выполнимой программы происходит переопределение переменной типа целый, то эта переменная и все связанные с ней переменные становятся неопределенными на втором уровне во всей этой основе.

В момент выполнения начального оператора основы переменная типа целый является определенной на втором уровне, если выполняются следующие условия:

в этой основе указанная переменная употребляется в индексе или вычисляемом операторе перехода;

в момент выполнения начального оператора этой основы указанная переменная определена на первом уровне.

10.2.8.2. Такое определение сохраняется до тех пор, пока не произойдет одно из следующих событий:

завершение выполнения конечного оператора этой основы;

указанная переменная становится неопределенной или переопределяется на первом уровне.

В этот момент указанная переменная становится неопределенной на втором уровне.

10.2.8.3. Вхождение переменной типа целый в список оператора ввода, в котором эта переменная употребляется также и в индексе, означает, что эта переменная станет определенной на втором уровне. Такое определение сохранится до тех пор, пока не произойдет одно из следующих событий:

завершение выполнения конечного оператора основы, содержащей этот оператор ввода;

указанная переменная становится неопределенной или переопределяется на первом уровне.

Переменная типа целый, использующаяся в качестве управляющей переменной списка с циклом, определена на втором уровне в области действия этого списка и только в этой области.

10.2.9. **Некоторые объекты модулей-функций.** Если в пределах одного оператора встречается более одного обращения к некоторому модулю-функции с одним и тем же списком фактических параметров, то выполнение этого модуля должно давать для всех этих случаев одни и те же результаты независимо от способа выполнения оператора.

Если оператор содержит множитель, который можно не вычислять (п. 6.4), а в этом множителе содержится ссылка на функцию, то все объекты, которые могли бы быть определены при выполнении этой ссылки на функцию, становятся неопределенными при завершении вычисления выражения, содержащего этот множитель.

10.3. **Требования определенности используемых объектов.** Любая переменная, употребляемая в индексе или вычисляемом операторе перехода, должна быть определена на втором уровне в момент ее использования.

Любая переменная, элемент массива или ссылка на функцию, употребляемые в качестве первичного выражения, а также любая ссылка на подпрограмму при помощи оператора вызова подпрограммы должны быть определены в момент их использования. В том случае, когда фактический параметр в списке параметров ссылки на внешнюю процедуру является именем переменной или именем элемента массива, на этот фактический параметр не накладывается требование определенности в момент ссылки на эту процедуру; однако если такой параметр является именем внешней процедуры, он должен быть определен.

Любая переменная, употребляемая в качестве начального параметра, конечного параметра или параметра приращения в операторе цикла или списке с циклом, должна быть определена в момент ее использования.

Любая переменная, употребляемая для идентификации устройства ввода/вывода, должна быть определена в момент ее использования.

В момент выполнения оператора возврата в модуле-функции значение (п. 8.3.1) этой функции должно быть определено.

В момент выполнения оператора вывода каждый объект, значение которого передается вовне, должен быть определен, за исключением случая, когда происходит вывод форматной записи (или ее части) и код преобразования есть A (то есть, вывод текста). Для остальных случаев вывода форматных записей требуется правильная связь кода преобразования с типом объекта. Следующие связи являются правильными: I — с типом целый; D — с типом двойной точности; E, F и G — с типами вещественный и комплексный; L — с типом логический.

(Измененная редакция, Изм. № 1).

АЛФАВИТНЫЙ УКАЗАТЕЛЬ ТЕРМИНОВ