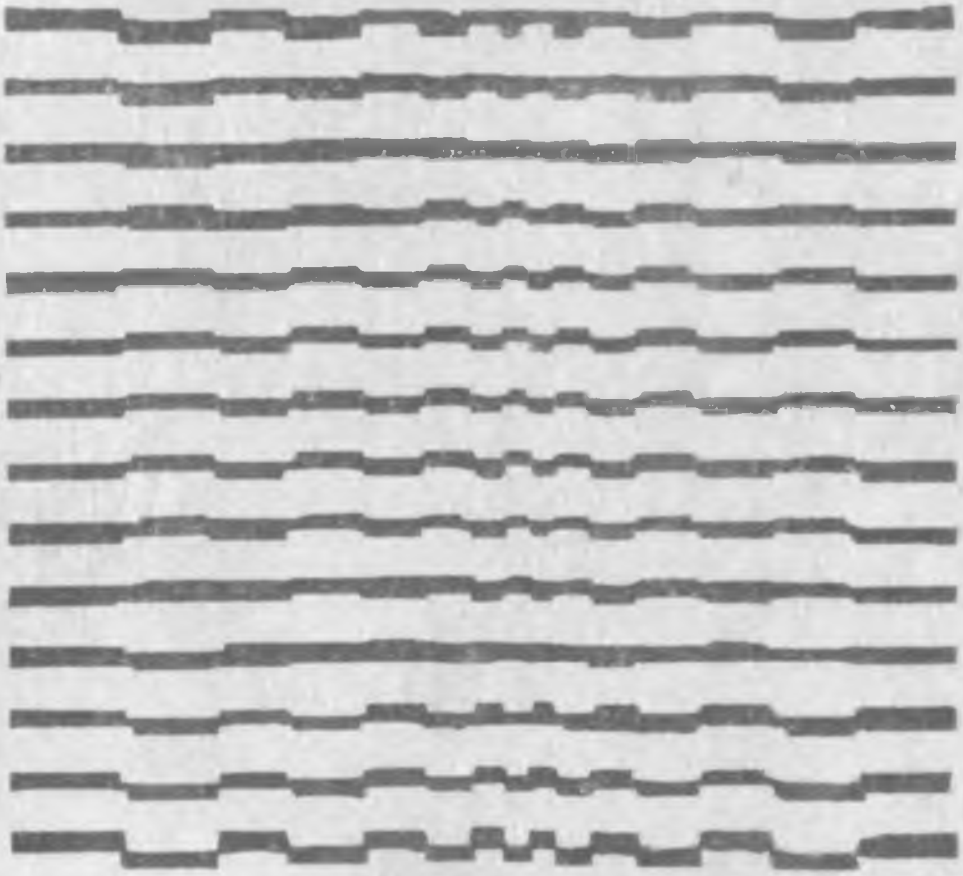
А. М. БУХТИЯРОВ
Ю. П. МАЛИНОВА
Г. Д. ФРОЛОВ

Практикум

по программированию на фортране



А. М. БУХТИЯРОВ,
Ю. П. МАЛИКОВА,
Г. Д. ФРОЛОВ

ПРАКТИКУМ ПО ПРОГРАММИРОВАНИЮ НА ФОРТРАНЕ

(ОС ЕС ЭВМ)

Под редакцией Н. А. КРИНИЦКОГО

ИЗДАНИЕ ВТОРОЕ, СТЕРЕОТИПНОЕ

Допущено Министерством высшего и среднего специального образования СССР в качестве учебного пособия для студентов высших учебных заведений

МОСКВА «НАУКА»
ГЛАВНАЯ РЕДАКЦИЯ
ФИЗИКО-МАТЕМАТИЧЕСКОЙ ЛИТЕРАТУРЫ
1993

22.18
Б 94
УДК 519.6

Практикум по программированию на фортране (ОС ЕС ЭВМ). Бухтияров А. М., Маликова Ю. П., Фролов Г. Д. — М.: Наука. Главная редакция физико-математической литературы, 1983.

Книга содержит описание алгоритмического языка фортран-IV, учитывающего ряд особенностей ЕС ЭВМ, сведения по операционной системе этих машин, необходимые для подготовки этих программ к их выполнению на ЭВМ.

В конце разделов описания языка содержится набор контрольных вопросов и совокупность задач для закрепления изучаемого материала. Ответы на эти задачи содержатся в конце книги.

Книга рассчитана на студентов вузов и втузов и может быть использована для обучения слушателей курсов профессионально-технической подготовки, а также для самостоятельного изучения фортрана.

Первое издание вышло в 1979 г.

Алексей Михайлович Бухтияров, Юлия Петровна Маликова, Геннадий Дмитриевич Фролов

ПРАКТИКУМ ПО ПРОГРАММИРОВАНИЮ НА ФОРТРАНЕ

Редакторы: *О. Ю. Маркадер, Е. Ю. Ходан*
Техн. редактор *Л. В. Лихачева*
Корректоры: *Т. С. Вайсберг, Н. Б. Румянцева*

ИБ № 12483

Печать с матриц. Подписано к печати 10.08.83.
Бумага 84 × 108¹/₃₂. тип. № 3. Обыкновенная гарнитура. Высокая печать.
Услови. печ. л. 15,96. Уч.-изд. л. 17,1. Тираж 85 000 экз. Заказ № 188.
Цена 75 коп.

Издательство «Наука»
Главная редакция физико-математической литературы.
117071, Москва, В-71, Ленинский проспект, 15
Отпечатано с матриц 2-й типографии издательства «Наука»
в Ленинградской типографии № 6 Ордена Трудового Красного Знамени
Ленинградского объединения «Техническая книга» им. Евгении Соколовой
Союзполиграфпрома при Государственном комитете СССР
по делам издательств, полиграфии и книжной торговли.
193144, г. Ленинград, ул. Моисеенко, 10.

Б $\frac{1702070000-139}{053\ (02)-83}$ 163-84

ОГЛАВЛЕНИЕ

ОТ РЕДАКТОРА

Методика программирования представляет собой главу прикладной математики, которую можно было бы назвать «Методы применения аппарата теории алгоритмов к решению различных задач». Прикладная математика предполагает использование определенного инструмента — в данном случае электронных вычислительных машин. В отличие от чистой математики (я не говорю — теоретической, потому, что прикладная математика тоже является теоретической), она требует не принципиальной выводимости результата из исходного данного и не его потенциальной вычислимости, а реальной возможности получения его, вычислимости с допустимым расходом ресурсов. В случае ЭВМ, таких ресурсов — три: общее время решения задачи, расходуемое машинное время и требуемый объем запоминающих устройств.

Предлагаемая читателю книга представляет собой руководство для решения определенной части вопросов, связанных с проблемой эффективной вычислимости при программировании на входном языке фортран. Самого знания этого формального языка еще не достаточно. Нужно овладеть еще определенной методикой программирования и рядом приемов такого управления трансляцией, которое обеспечило бы нужные свойства программы. В учебниках и задачниках по применению программирования на входном языке фортран, которые выходили до сих пор, эта сторона работы проблемного программиста оставалась в стороне. Оставалась в стороне и связь программы с операционной системой ЭВМ (главной программой системы ее математического обеспечения) и возможность влиять на эту связь.

Не вызывает сомнения, что данный Практикум принесет пользу не только при обучении программированию, но и при работе уже обученных программистов и кое в чем окажется интересным даже для знатоков.

Н. А. Криницкий

ПРЕДИСЛОВИЕ

В последнее время в нашей стране широкое распространение получили вычислительные машины единой системы ЭВМ (ЕС ЭВМ), относящиеся к ЭВМ третьего поколения. Они имеют весьма развитое программное обеспечение, основу которого составляют операционные системы. Операционные системы — это комплексы программных средств, под управлением которых осуществляется выполнение на ЭВМ программ решения конкретных задач (прикладных программ). Операционные системы позволяют:

а) увеличить пропускную способность вычислительной системы (за счет обеспечения постоянной занятости всех ее компонент);

б) уменьшить время реакции вычислительной системы (за счет уменьшения доли участия человека в управлении потоком выполнения прикладных программ);

в) упростить разработку прикладных программ (за счет использования большого количества эффективных языков программирования, большого набора обслуживающих программ, возможности накапливания библиотек прикладных программ и т. д.).

Комплекс программ, входящий в состав любой операционной системы, можно разделить на две части: на управляющую программу и на обслуживающие программы.

В функции управляющей программы входят планирование прохождения потока прикладных программ через ЭВМ, распределение машинных ресурсов между прикладными программами, организация мультипрограммной работы, устранение последствий сбоев в работе оборудования и т. д.

В функции обслуживающих программ входит перевод прикладных программ с языков программирования на язык ЭВМ, обеспечение отладки прикладных программ, обслуживание библиотек программ и наборов данных и т. д.

Таким образом, программирование для ЕС ЭВМ требует знаний не только языков программирования, но и знаний возможностей операционных систем по подготовке прикладных программ к выполнению на ЭВМ и по управлению процессом их выполнения на ЭВМ.

К наиболее распространенным операционным системам ЕС ЭВМ относится операционная система, известная под названием ОС ЕС ЭВМ. Она может использоваться на большинстве моделей ЕС ЭВМ, обладает большими возможностями по организации вычислительного процесса и рассчитана на самые разнообразные применения. ОС ЕС ЭВМ имеет многоязыковую систему программирования, в состав которой входят машинно-ориентированный язык (язык ассемблера), и ряд алгоритмических языков, среди которых наибольшей популярностью пользуется фортран.

Настоящая книга преследует цель привить практические навыки в составлении программ на языке фортран, ориентированном на операционную систему ОС ЕС ЭВМ. Книга состоит из 10 глав.

Главы с 1-й по 8-ю содержат детальное описание конструкций языка фортран, которое дается с учетом понятий и терминологии, принятых в ОС ЕС ЭВМ.

Главы 9-я и 10-я содержат ряд сведений об операционной системе ОС ЕС ЭВМ, необходимых для перевода программ с языка фортран на машинный язык, для подготовки заданий на выполнение программ на ЭВМ и т. д.

Главы делятся на разделы, которые содержат много примеров и, как правило, заканчиваются перечнями вопросов и упражнений. На некоторые из них даются ответы, которые приведены в конце книги.

Для изучения материала не требуется предварительных знаний ни по языку фортран, ни по операционной системе ОС ЕС ЭВМ. Книга рассчитана на студентов вузов и научных сотрудников.

А. М. Бухтияров,
Ю. П. Маликова,
Г. Д. Фролов

Глава 1

ОСНОВНЫЕ ЭЛЕМЕНТЫ ЯЗЫКА

1.1. Структура программ

Первичными элементами языка фортран, из которых строятся все его конструкции, являются символы. По своей роли в программах их можно разделить на два набора: основных символов и дополнительных символов. *Набор основных символов* строго фиксирован и используется для образования всех конструкций языка. *Набор дополнительных символов* в языке не фиксирован. Он определяется составом оборудования ЭВМ и используется для образования символьных констант, а также для представления данных на носителях информации.

Основными конструкциями языка являются операторы. Они делятся на два класса: на выполняемые и невыполняемые. *Выполняемые операторы* используются в программах для указания действий и порядка выполнения этих действий. *Невыполняемые операторы* используются для указания обозначений величин, для указания их типа и структуры, для задания информации о распределении памяти и т. д.

Программа на языке фортран может состоять либо из одной *основной программы*, либо из одной основной программы и нескольких подпрограмм. Выполнение программы всегда начинается с основной программы. Подпрограммы подключаются к работе путем обращения к ним либо из основной программы, либо из других подпрограмм.

Основную программу и подпрограммы принято называть *программными модулями*.

Пример структуры программы, изображенной в виде схемы, приведен на рис. 1.1, где сплошными стрелками показаны возможные обращения одних программных модулей к другим, а пунктирными стрелками — возвраты, следующие после выполнения соответствующих программных модулей.

Программа, написанная на языке фортран, превращается в машинную программу (в программу на машинном языке) путем ее переработки специальными системными программами, к которым относятся *транслятор* и *редактор связей*.

Транслятор используется для перевода программных модулей на промежуточный язык, называемый языком *объектных модулей*.

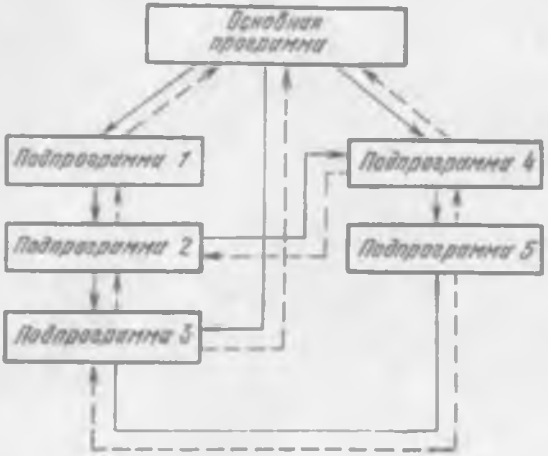


Рис. 1.1. Пример структуры программы.

Редактор связей используется для объединения (сборки) объектных модулей в единую программу, называемую загрузочным модулем.

Программу на фортране иногда называют *фортран-программой*, а эквивалентную ей программу на машинном языке (*загрузочный модуль*) — рабочей программой.

Загрузочный модуль может содержать один или несколько сегментов (иметь простую или оверлейную структуру). В последнем случае один из сегментов должен быть *корневым*. В состав корневого сегмента должна входить основная программа. Кроме того, в него может включаться часть подпрограмм. Остальные (не корневые) сегменты должны состоять только из подпрограмм.

Загрузочные модули размещаются редактором связей в специальные библиотеки, которые создаются на внешних запоминающих устройствах прямого доступа (на магнитных дисках или на магнитных барабанах).

Корневые сегменты программ вызываются для работы в оперативную память из библиотек загрузочных модулей с помощью управ-

ляющих операторов, входящих в состав так называемых заданий на работу. Задание на работу состоит из управляющих операторов, с помощью которых операционная система ЭВМ получает сведения о том, какое имя имеет загрузочный модуль, в какой из библиотек он содержится, на каком из внешних запоминающих устройств размещаются библиотека программ и данные, необходимые для выполнения данного загрузочного модуля и т. д.

Не корневые сегменты вызываются для работы в оперативную память при обращении к подпрограммам, входящим в состав этих сегментов. Такие сегменты в процессе работы программы могут перекрывать друг друга (размещаться в одни и те же места оперативной памяти) и, следовательно, вызываться в оперативную память по нескольку раз.

Структура загрузочного модуля (количество сегментов, их состав, характер перекрытия сегментов и т. д.) планируется разработчиком программы. Информация о ней сообщается редактору связей с помощью управляющих операторов и используется при объединении объектных модулей в загрузочный. Описание языка, на котором представляются задания на работу (языка управления заданиями), приведено в гл. 9.

Вопросы и упражнения

1. На какие классы делятся операторы в языке фортран?
2. Какую роль в программе играют невыполняемые операторы?
3. Что такое программный модуль и из каких компонент он состоит?
4. Что такое рабочая программа?
5. Описать, какие этапы обработки должна пройти фортран-программа для получения рабочей программы и какие специальные системные программы для этого используются?
6. Какую структуру может иметь рабочая программа?
7. Где размещаются загрузочные модули?

1.2. Основные символы

Основные символы языка делятся на следующие группы: *буквы*, *цифры* и *специальные символы*. В качестве букв используются двадцать шесть букв латинского алфавита:

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.

Цифра — это один из следующих десяти символов: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9. Символ 0 означает нуль. В рукописных текстах нуль

Таблица 1.1. Специальные символы

Символ	Название символа	Символ	Название символа
—	Пробел	(	Открытая скобка
=	Равно	)	Закрытая скобка
+	Плюс	,	Запятая
−	Минус	.	Точка
*	Звездочка	&	Амперсенд
/	Косая черта	'	Апостроф

перечеркивается, что дает возможность всегда отличать его от буквы О. Специальный символ — это один из символов, перечисленных в табл. 1.1.

В дальнейшем под *буквенно-цифровым символом* будем понимать букву или цифру из набора основных символов.

Из символов языка образуются слова данного языка, которые являются минимальными конструкциями, имеющими в данном языке некоторый самостоятельный смысл. Эти слова могут обозначать числа, имена величин, некоторые действия и т. д.

В языке фортран различаются два типа слов: *служебные слова* и *слова пользователя*. Служебные слова имеют фиксированное начертание и раз и навсегда заданный смысл. Слова пользователя выбираются разработчиком программы по его усмотрению, но с соблюдением определенных правил, принятых в данном языке.

Слова пользователя по своему начертанию не должны совпадать со служебными словами.

Служебные слова будут приводиться в процессе описания тех конструкций языка, в которых они могут встречаться.

Вопросы и упражнения

1. Какое количество основных символов используется в языке фортран?

2. Входят ли в состав основных символов строчные латинские буквы *a*, *b*, *c*, *d*, . . . , русские буквы, синтаксические знаки русского языка: точка, запятая, точка с запятой, двоеточие?

3. Какие наборы символов можно использовать при написании фортран-программы?

4. На какие типы подразделяются слова в языке фортран?

1.3. Типы величин

В программах на языке фортран могут использоваться величины следующих типов: *целого*, *вещественного*, *комплексного*, *логического* и *символьного*.

Значениями целых величин могут быть целые положительные числа, целые отрицательные числа и число нуль.

Значениями вещественных величин являются действительные числа. Они могут быть положительными, отрицательными и равными нулю.

Значением комплексной величины является пара действительных чисел. Первое из этих чисел означает действительную часть комплексной величины, а второе — коэффициент при мнимой части.

Логические величины могут принимать только логические значения истина и ложь.

В качестве значений величин символьного типа могут выступать последовательности символов из основного и дополнительного наборов.

Для величин целого, вещественного, комплексного и логического типов предусмотрены две длины: *стандартная* и *нестандартная*.

Длина целой величины характеризует собой диапазон значений этой величины. Целая величина стандартной длины может принимать значения от $-(2^{31}-1)$ до $+(2^{31}-1)$, т. е. от $-2\,147\,483\,647$ до $+2\,147\,483\,647$. В оперативной памяти ЭВМ значение такой величины занимает четырехбайтовую ячейку и представляется в виде двоичного числа с фиксированной точкой.

Целая величина нестандартной длины может принимать значения от $-(2^{15}-1)$ до $+(2^{15}-1)$, т. е. от $-32\,767$ до $+32\,767$. В памяти ЭВМ значение такой величины занимает двухбайтовую ячейку и представляется в виде двоичного числа с фиксированной точкой.

Длина вещественной величины характеризует точность представления значения этой величины. Значение вещественной величины стандартной длины представляется с точностью до семи десятичных знаков. В памяти ЭВМ оно хранится в четырехбайтовой ячейке в виде двоичного числа с плавающей точкой. Значение вещественной величины нестандартной длины представляется с точностью до шестнадцати десятичных знаков. В памяти ЭВМ оно хранится в восьмибайтовой ячейке в виде двоичного числа с плавающей точкой. Вещественные величины могут принимать значения в диапазоне от -10^{75} до $+10^{75}$.

Длина комплексной величины характеризует точность представления значений действительной части и коэффициента при мнимой части этой величины таким же образом, как и в случае вещественных величин. Под значение комплексной величины стандартной длины отводятся две четырехбайтовые ячейки, а нестандартной длины — две восьмибайтовые ячейки. Эти значения хранятся в виде двоичных чисел с плавающей точкой.

Под значение логической величины стандартной длины отводится четырехбайтовая ячейка, а нестандартной длины — однобайтовая ячейка. Значение истина изображается в виде последовательности битов, состоящей из нулей и одной единицы (в последнем разряде справа). Значение ложь — в виде последовательности битов, состоящей из нулей.

Объем оперативной памяти, отводимый для размещения символьных величин, зависит от особенностей тех конструкций языка, в состав которых эти величины входят, и будет указан в описаниях этих конструкций.

Величины целого, вещественного, комплексного и логического типов могут использоваться в фортран-программах в виде констант, переменных и массивов различной размерности (векторов, матриц и т. д.). Величины символьного типа — только в виде констант.

Вопросы и упражнения

1. Какие типы величин используются в языке фортран?
2. Чем можно объяснить наличие двух длин значений переменных?
3. Указать диапазон значений величин целого и вещественного типов

1.4. Константы

Константы используются в фортран-программах для представления постоянных значений величин (чисел, логических значений и т. д.). Различаются шесть типов констант: целые, вещественные, комплексные, логические, символьные и шестнадцатеричные. По внешнему виду написания константы можно определить ее тип и значение.

1.4.1. Целые константы. *Целые константы* используются для представления целых десятичных чисел и записываются в следующем виде:

$$sa_1a_2 \ldots a_n$$

где s — либо пусто (пусто означает отсутствие какого-либо символа), либо знак числа (плюс или минус); a_i ($i = 1, 2, \ldots, n$) — цифры. Целые константы без знака и со знаком + (плюс) означают целые положительные числа и число нуль, а со знаком — (минус) — целые отрицательные числа.

Примеры целых констант:

0 (означает число 0),

15 (означает число 15),

+35 (означает число 35),

—1248 (означает число — 1248).

Целые константы считаются величинами целого типа стандартной длины. Под каждую из них отводится четырехбайтовая ячейка.

Вопросы и упражнения

1. С какой целью в фортран-программах используются целые константы?

2. Что означают целые константы без знака, со знаком плюс и со знаком минус?

3. Как различаются целые константы по длине?

4. Сколько байтов оперативной памяти отводится для записи целой константы?

5. Представить приведенные ниже числа в виде целых констант:

1) $-32 \cdot 2^5$ 4) $1/2^{-12}$ 7) $26{,}3 \cdot 10^3$
2) $2/25 \cdot 10^5$ 5) —0 8) $0{,}019 \cdot 10^6$
3) $0{,}0 \cdot 7^8$ 6) 00647 9) +0

6. Указать, какие из приведенных ниже целых констант определяют одно и то же число:

1) 000641 4) +0 7) —000512
2) —0000000 5) 641 8) +0641
3) —512 6) 00001 9) —00641

1.4.2. Вещественные константы. *Вещественные константы* используются в фортран-программах для представления действительных чисел. Они могут записываться в двух видах: без экспоненты (так называемая форма F) и с экспонентой (так называемые формы E и D).

Вещественная константа без экспоненты может быть представлена в одном из следующих видов:

а) $sa_1a_2 \ldots a_n.b_1b_2 \ldots b_k$

б) $sa_1a_2 \ldots a_n.$

в) $s.b_1b_2 \ldots b_k$

Здесь s — либо пусто, либо знак числа (плюс или минус); a_i ($i = 1, 2, \ldots, n$) — цифры целой части числа; b_j ($j = 1, 2, \ldots, k$) — цифры дробной части числа. Символ точка указывает конец целой части (или же начало дробной части).

Константы без знака и со знаком плюс означают положительные числа и число нуль, а со знаком минус — отрицательные числа.

Примеры вещественных констант без экспонент:

0.0 (означает число нуль),

.125 (означает число 0,125),

—.3156 (означает число —0,3156),

+95.46 (означает число 95,46),

—13450. (означает число —13450,0).

Следует особо подчеркнуть тот факт, что вещественная константа без экспоненты обязательно должна содержать точку (ее принято называть десятичной точкой). Десятичная точка может стоять в начале, в середине и в конце константы. Вещественная константа без экспоненты относится к величинам вещественного типа стандартной длины и занимает в оперативной памяти ЭВМ четырехбайтовую ячейку.

Представление числа в виде вещественной константы без экспоненты (или в форме F) удобно использовать, если это число содержит небольшое количество цифр.

Вещественная константа с экспонентой E имеет вид:

$$k_1Ek_2$$

где k_1 — вещественная константа без экспоненты называемая мантиссой, k_2 — целая константа, содержащая не более двух цифр и называемая *порядком*. Конструкция Ek_2 называется *десятичной экспонентой* и представляет собой множитель 10^{k_2}. Следовательно, константа k_1Ek_2 означает число, равное произведению $k_1 \cdot 10^{k_2}$.

Примеры вещественных констант с экспонентой типа E:

3.5E+3 (означает число $3{,}5 \cdot 10^3$),

—.5 E — 05 (означает число $-0{,}5 \cdot 10^{-5}$),

14.E0 (означает число 14,0),

1347591.E18 (означает число $1347591{,}0 \cdot 10^{18}$).

Вещественная константа с экспонентой типа E относится к величинам вещественного типа стандартной длины и занимает в памяти ЭВМ четырехбайтовую ячейку.

Вещественная константа с экспонентой типа D имеет вид:

$k_1 D k_2$

где k_1 — вещественная константа без экспоненты, называемая мантиссой; k_2 — целая константа, содержащая не более двух цифр, называемая порядком. Как и в предыдущем случае, конструкция $D k_2$ является десятичной экспонентой (множителем 10^{k_2}). Следовательно, константа $k_1 D k_2$ означает число, равное произведению $k_1 \cdot 10^{k_2}$.

Примеры вещественных констант с экспонентой типа D:

3.5D+3 (означает число $3{,}5 \cdot 10^3$),

—.5D — 05 (означает число $-0{,}5 \cdot 10^{-5}$),

5348974.15D3 (означает число $5348974{,}15 \cdot 10^3$).

Вещественная константа с экспонентой D относится к величинам вещественного типа нестандартной длины и занимает в памяти ЭВМ восьмибайтовую ячейку.

Следует особо подчеркнуть тот факт, что при написании констант с экспонентой E или D ни мантиссу, ни порядок нельзя опускать. Если мантисса или порядок положительные числа, то знак перед ним может быть опущен.

Форма E и D удобны для представления больших и малых чисел.

Вопросы и упражнения

1. С какой целью в фортран-программах используются вещественные константы?

2. Сколько различают видов вещественных констант и какие?

3. В каком случае для представления числа удобно использовать вещественную константу без экспоненты и с экспонентой?

4. Можно ли при написании вещественной константы с экспонентой опустить нулевой порядок?

5. Может ли в вещественной константе порядок быть константой из трех цифр?

6. Можно ли в вещественной константе +6420. опустить десятичную точку?

7. Как различаются вещественные константы без экспоненты по длине?

8. Сколько байтов оперативной памяти используется для представления вещественной константы без экспоненты?

9. Сколько байтов оперативной памяти используется для представления вещественной константы типа D?

10. Представить приведенные ниже числа в виде вещественных констант: а) без экспоненты стандартной и нестандартной длины, б) с экспонентой типа E и D:

1) -0 4) 0 7) 3^{-2}
2) $264{,}1\cdot10^{-2}$ 5) -1 8) $1/25$
3) $0{,}9999\cdot10^{2}$ 6) $0{,}01$ 9) $+0{,}0$

11. Указать, какие из приведенных ниже вещественных констант определяют одно и то же число:

1) —24.001 6) —2.400E1 11) .000
2) 0. 7) —0.00E—1 12) —.301D01
3) —0.24001E+2 8) —3.01D2 13) —.24001D+2
4) 0.001 9) 2.40010E1 14) .1E—2
5) 0.D—00 10) —30.100 15) .0D—00

Какие из приведенных констант имеют стандартную длину? Какие из этих констант могут быть отнесены к константам типа F?

1.4.3. Комплексные константы. *Комплексные константы* используются для представления комплексных чисел. Комплексная константа представляет собой упорядоченную пару вещественных констант и имеет вид: (a_1, a_2), где a_1 — вещественная константа, означающая действительную часть комплексного числа; a_2 — вещественная константа, означающая коэффициент при мнимой части комплексного числа.

Обе компоненты комплексной константы a_1 и a_2 должны иметь одинаковые длины. Комплексная константа, компоненты которой имеют стандартные длины, относится к величинам комплексного типа стандартной длины и занимает в памяти ЭВМ две четырехбайтовые ячейки. Комплексная константа, компоненты которой имеют нестандартные длины, относится к величинам комплексного типа нестандартной длины и занимает в памяти ЭВМ две восьмибайтовые ячейки.

Примеры комплексных констант:

(3.75,—1.15E2) (означает число $3{,}75-115\,i$),

(5.D — 5,1.5D0) (означает число $5\cdot10^{-5}+1.5\,i$),

(.5,+13.) (означает число $0{,}5+13\,i$),

(0.1E5,—3.0E12) (означает число $10^{4}-3\cdot10^{12}\,i$),

(1.0D15,7.0D — 3) (означает число $10^{15}+7\cdot10^{-3}\,i$),

(745.894D6,0.0000012537D0) (означает число $745{,}894\cdot10^{6}+12{,}537\cdot10^{-7}\,i$).

Константы, приведенные в первом, третьем и четвертом примерах, относятся к комплексным константам стандартной длины, а остальные — к комплексным константам нестандартной длины.

Вопросы и упражнения

1. С какой целью используются в фортран-программе комплексные константы?

2. Как различаются комплексные константы по длине?

3. Допустимо ли написание в качестве компонент a_1 и a_2 комплексной константы (a_1, a_2), вещественных констант разной длины (a_1 — стандартной длины и a_2 — нестандартной длины или a_1 — нестандартной длины, а a_2 — стандартной длины)?

4. Сколько байтов оперативной памяти потребуется для представления комплексной константы стандартной длины?

5. Представить приведенные ниже числа в виде комплексных констант а) стандартной длины, б) нестандартной длины:

1) $-3{,}6i$
2) $0{,}649$
3) $-3{,}7-10^3i$
4) $0{,}3+2{,}9i$
5) $6{,}99-3{,}14\cdot10^3i$
6) $3-4i$
7) $29{,}9i$
8) $-14{,}7i^3$
9) $-6{,}999$

6. Какие из приведенных ниже комплексных констант определяют одно и то же число:

1) (—0.,+2.4)
2) (6.3, —4.1E—2)
3) (—.0E00,24.E—1)
4) (—0.E—2,0.0024E3)
5) (6.3D0,—4.1D—2)
6) (0.63E1,—.0041E + 1)
7) (—.0,+.24E01)
8) (.63D—1,—.41D—1)
9) (6.3.E—1,0.041)

1.4.4. **Логические константы.** *Логические константы*, означающие логические значения истина и ложь, записываются как последовательности символов .TRUE. и .FALSE. соответственно.

Отсутствие хотя бы одной точки (справа или слева) в написании логической константы воспринимается как ошибка. Логические константы относятся к величинам логического типа стандартной длины и каждая из них занимает в памяти четырехбайтовую ячейку.

Вопросы и упражнения

1. С какой целью используются в фортран-программе логические константы?

2. Правильно ли на языке фортран записано значение истина как TRUE.?

3. К какому типу величин относятся логические константы?

4. Сколько байтов оперативной памяти необходимо для записи логической константы .FALSE.?

5. Как представляется логическая константа .TRUE. в оперативной памяти?

1.4.5. Символьные константы. *Символьные константы* используются в программах для указания начальных значений величин, в качестве фактических параметров (при обращении к подпрограммам), а также в операторах формата. Они могут записываться в двух формах: с указателем длины и без указателя длины.

Символьные константы с указателем длины имеют вид:

*n*H*s*

где *s* — последовательность символов, представляющая собой значение символьной константы, *n* — целая константа без знака, означающая число символов в последовательности *s*; H — буква, служащая разделителем между константой и последовательностью символов *s*. Последовательность *s* может содержать символы как из основного так и из дополнительного наборов.

Каждый символ из последовательности *s* представляется в основной памяти в виде восьмиразрядного двоичного кода в соответствии с табл. 1.2. Так, например, символьная константа 3HFOR будет представлена в виде последовательности восьмиразрядных двоичных кодов символов FOR, т. е. в виде последовательности битов 110001101101011011011001.

Конструкция *n*H называется *указателем длины символьной константы.*

Символьные константы без указателя длины имеют вид:

'*s*' где *s* — последовательность символов.

В этом случае апостроф внутри последовательности *s* обозначается двумя апострофами. Так, например, символьная константа 'A''105''' будет представлена в виде последовательности восьмиразрядных двоичных кодов следующих символов: A'105'.

Пробел внутри последовательности *s* рассматривается как символ.

Количество символов в последовательности *s* не должно превышать числа 255. Символьные константы размещаются в ячейках памяти таким образом, чтобы каждый из символов последовательности *s* попал в отдельную однобайтовую ячейку. Таким образом, для представления начального значения однобайтовой величины требуется один символ, двухбайтовой величины — два символа и т. д.

Таблица 1.2. Коды символов ЕС ЭВМ

Символ	Двоичный код символа	16-ричный код символа	Символ	Двоичный код символа	16-ричный код символа
A	11000001	C1	a	10000001	81
B	11000010	C2	b	10000010	82
C	11000011	C3	c	10000011	83
D	11000100	C4	d	10000100	84
E	11000101	C5	e	10000101	85
F	11000110	C6	f	10000110	86
G	11000111	C7	g	10000111	87
H	11001000	C8	h	10001000	88
I	11001001	C9	i	10001001	89
J	11010001	D1	j	10010001	91
K	11010010	D2	k	10010010	92
L	11010011	D3	l	10010011	93
M	11010100	D4	m	10010100	94
N	11010101	D5	n	10010101	95
O	11010110	D6	o	10010110	96
P	11010111	D7	p	10010111	97
Q	11011000	D8	q	10011000	98
R	11011001	D9	r	10011001	99
S	11100010	E2	s	10100010	A2
T	11100011	E3	t	10100011	A3
U	11100100	E4	u	10100100	A4
V	11100101	E5	v	10100101	A5
W	11100110	E6	w	10100110	A6
X	11100111	E7	x	10100111	A7
Y	11101000	E8	y	10101000	A8
Z	11101001	E9	z	10101001	A9
Б	10111010	BA	б	01111000	78
Г	10111111	BF	г	10001101	8D
Д	10111100	BC	д	10001010	8A
Ж	11101100	EC	ж	10101110	AE
З	11111010	FA	з	10110010	B2
И	11001011	CB	и	10001111	8F
Й	11001100	CC	й	10010000	90
Л	11001110	CE	л	10011011	9B
П	11011100	DC	к	10011010	9A

Таблица 1.2 (продолжение)

Символ	Двоичный код символа	16-ричный код символа	Символ	Двоичный код символа	16-ричный код символа
У	11101011	EB	м	10011100	9C
Ф	10111110	BE	н	10011101	9D
Ц	10111011	BB	п	10011111	9F
Ч	11111110	FE	т	10101100	AC
Ш	11111011	FB	у	10101000	A8
Щ	11111101	FD	ф	10001100	8C
Э	11111100	FC	ц	10000000	80
Ы	11101111	EF	ч	10110110	B6
Ь	11101110	EE	ш	10110011	B3
Ю	10111000	B8	щ	10110101	B5
Я	11011101	DD	э	10110100	B4
—	01000000	40	ы	10110001	B1
&	01010000	50	ь	10110000	B0
—	01100000	60	ъ	10110111	B7
{	11000000	C0	ю	01110110	76
}	11010000	D0	я	10100000	A0
/	01100001	61	`	01111101	7D
'	01111001	79	+	01001110	4E
[	01001010	4A	:	01011110	5E
]	01011010	5A	>	01101110	6E
|	01101010	6A	=	01111110	7E
\	11100000	E0	!	01001111	4F
:	01111010	7A	—	01011111	5F
.	01001011	4B	?	01101111	6F
¤	01011011	5B	»	01111111	7F
,	01101011	6B	0	11110000	F0
#	01111011	7B	1	11110001	F1
<	01001100	4C	2	11110010	F2
*	01011100	5C	3	11110011	F3
%	01101100	6C	4	11110100	F4
@	01111100	7C	5	11110101	F5
(	01001101	4D	6	11110110	F6
)	01011101	5D	7	11110111	F7
—	01101101	6D	8	11111000	F8
			9	11111001	F9

Вопросы и упражнения

1. С какой целью используются символьные константы в фортран-программе?

2. Может ли значение символьной константы не содержать ни одного символа?

3. Может ли значение символьной константы состоять только из букв русского алфавита?

4. Какое минимальное и максимальное количество символов может содержаться в значении символьной константы?

5. Указать, какие из приведенных ниже записей можно рассматривать как символьные константы:

1) 3HOA+B/2
2) '4.3—F"*B'''
3) '34+A'B
4) 7H.FALSE.
5) 'STOP"A—B''''
6) ''''''
7) —2HAB
8) '2HAB'
9) .'TPUE'.
10) 3H—2

6. Представить приведенные ниже записи в виде символьных констант: а) с указателем длины, б) без указателя длины:

1) —0,64201
2) ABCDE—2
3) (—48+3*A)*B
4) 6A"B'—C
5) ,+,—,*,
6) 2—A**2
7) 3HF—V
8) A'B"C'''
9) ''''
10) ((((

7. Указать, как будут представлены в памяти машины приведенные ниже символьные константы:

1) 4HROFA
2) 'SmTrt'
3) '_Z+A'''
4) 3H128
5) '—f0'
6) 2H3%

1.4.6. Шестнадцатеричные константы. *Шестнадцатеричные константы* используются в программе только для указания начальных значений величин. Они имеют вид:

$$Za_1a_2 \ldots a_n$$

где Z — основной символ (буква), a_i ($i = 1, 2, \ldots, n$) — шестнадцатеричные цифры.

В языке фортран шестнадцатеричные цифры принято изображать следующими символами основного набора: 0, 1, 2, 3, 4, 5, 6, 7,

Таблица 1.3. Двоичные коды шестнадцатеричных цифр

Цифра	Код	Цифра	Код	Цифра	Код	Цифра	Код
0	0000	4	0100	8	1000	C	1100
1	0001	5	0101	9	1001	D	1101
2	0010	6	0110	A	1010	E	1110
3	0011	7	0111	B	1011	F	1111

8, 9, A, B, C, D, E, F. Каждая шестнадцатеричная цифра константы в основной памяти представляется в виде четырехразрядного двоичного кода в соответствии с табл. 1.3.

Примеры шестнадцатеричных констант:

Z0 (означает последовательность битов 0000),

ZF (означает последовательность битов 1111),

Z1A (означает последовательность битов 0001 1010).

Шестнадцатеричные константы размещаются в ячейках памяти таким образом, чтобы каждая пара цифр попала в отдельную однобайтовую ячейку. Из этого следует, что для представления начального значения однобайтовой величины требуется две шестнадцатеричные цифры, двухбайтовой — четыре шестнадцатеричных цифр и т. д.

Примеры начальных значений величин, представленных шестнадцатеричными константами, приведены в табл. 1.4.

Таблица 1.4. Примеры шестнадцатеричных констант

	Тип и длина величины	Значение величины	Шестнадцатеричная константа
1	Целая нестандартной длины	31	Z001F
2	Целая стандартной длины	31	Z0000001F
3	Логическая нестандартной длины	Ложь	Z00
4	Логическая стандартной длины	Истина	Z00000001

1.5. Идентификаторы

Идентификаторы используются в программах для обозначения переменных, массивов, функций, процедур, общих блоков и т. д. Идентификатор представляет собой последовательность букв и цифр из основного алфавита, причем первым символом этой последовательности должна быть буква. В идентификаторе должно быть не более шести символов.

Примеры идентификаторов:

A
SIN
B15
TIME23

Последовательности символов 2A36 и A023BC76 не идентификаторы, так как в первом случае последовательность начинается с цифры, во втором — содержит количество символов больше шести.

Идентификаторы относятся к словам пользователя и выбираются разработчиком программ по своему усмотрению. Они не должны совпадать со служебными словами.

Вопросы и упражнения

1. С какой целью в фортран-программе используются идентификаторы?

2. Может ли идентификатор содержать девять символов и начинаться с цифры?

3. Может ли идентификатор содержать буквы Б и Ф русского алфавита?

4. Указать, какие из приведенных ниже записей можно рассматривать как идентификаторы:

1) AO21B	4) A+B = C	7) SIFA464
2) 3FAB	5) IRa	8) SAIRA
3) VA'CD	6) SUM	9) X1

5. Определить количество различных идентификаторов, которые можно образовать из а) символа X и б) символов A и 1.

6. Можно ли идентификатор рассматривать как служебное слово?

1.6. Переменные. Массивы

Переменная — это величина, которая может принимать различные значения. Различают простые переменные и переменные с индексами.

Простая переменная обозначается идентификатором. Например,

X1, Y35, LOG.

В процессе перевода фортран-программы на машинный язык простая переменная рассматривается как обозначение некоторого поля памяти (ячейки), содержимое которой является значением этой переменной в каждый момент времени вычислительного процесса.

Переменная с индексами представляется конструкцией

$$a\,(i_1, i_2, \ldots, i_n)$$

где a — идентификатор, i_k $(k = 1, 2, \ldots, n)$ — индекс, представляющий собой *арифметическое выражение* целого или вещественного типа (см. п. 1.8). Значение индекса всегда округляется до ближайшего целого. Таким образом, переменная с индексами, в которой хотя бы один из индексов отличен от константы, как и на математическом языке, представляет собой краткую запись упорядоченной совокупности переменных с индексами, выраженных константами. Например, переменная с индексом X (I, J) представляет краткую запись упорядоченной совокупности переменных с индексами:

X(2,1), X(2,2), X(2,3), X(2,4), X(3,1), X(3,2), X(3,3), X(3,4), если I = 2,3 и J = 1, 2, 3, 4.

Упорядоченная совокупность всех элементов a (1, 1, 1, . . . , 1), a (2, 1, 1, . . . , 1), . . . , a (T_1, 1, 1, . . . , 1), a (1, 2, 1, . . . , 1), a (2, 2, 1, . . . , 1), . . . , a (T_1, 2, 1, . . . , 1), a (1, 3, 1, . . . , 1), a (2, 3, 1, . . . , 1), . . . , a (T_1, 3, 1, . . . , 1), . . . , a (1, T_2, T_3, . . . , T_n), a (2, T_2, T_3, . . . , T_n), . . . , a (T_1, T_2, T_3, . . . , T_n), где a — идентификатор, называется n-мерным массивом. Количество элементов массива равно $T_1 \times T_2 \times T_3 \times \ldots \times T_n$.

Переменные и массивы могут относиться к целому, вещественному, комплексному и логическому типам и могут иметь стандартную и нестандартную длину.

Способы задания типа и длины простых переменных и массивов описываются в последующих главах. Переменные с индексами имеют тот же тип и ту же длину, что и массивы, элементами которых они являются.

Порядковый номер элемента внутри массива называется *приведенным индексом*. Понятие приведенного индекса часто используется в фортран-программах. Значение приведенного индекса для элемента двумерного массива можно вычислить по формуле $i_1 + T_1(i_2 - 1)$, а для трехмерного массива — по формуле $i_1 + T_1(i_2 - 1) + T_1 \cdot T_2(i_3 - 1)$. В этих формулах i_1, i_2 и i_3 означают значения первого, второго и

третьего индекса соответственно, а T_1, T_2 и T_3 — их максимальные значения.

Вопросы и упражнения

1. Какие переменные допустимы в фортран-программе?

2. Чем является простая переменная, заданная в фортран-программе, после перевода этой программы на машинный язык?

3. Можно ли запись A (X, 2, Y3) рассматривать как переменную с индексами, если X — переменная вещественного типа, а Y3 — комплексного типа?

4. Сколько элементов содержит трехмерный массив, если максимальное значение первого индекса равно 3, второго — 3 и третьего — 4?

5. Пусть максимальные значения первого и второго индексов массива X1 соответственно равны 5 и 3. Вычислить порядковые номера приведенных ниже элементов массива X1:

1) X1(3,2) 4) X1(2,3)
2) X1(1,2) 5) X1(1,1)
3) X1(5,3) 6) X1(4,2)

6. Рассматривая каждую из приведенных ниже матриц как двумерный массив, представить ее в виде последовательности переменных с индексами:

1) (a_{ij}) $(i = 1, 2, 3;\ j = 1, 2)$,
2) (a_{jk}) $(j = 1, 2;\ k = 1, 2, 3, 4)$,
3) (b_{ij}) $(i = 1, 2;\ j = 1, 2, 3)$.

1.7. Указатели функций

В языке фортран с понятием функции связано два понятия — это *указатель функции* и *подпрограмма* (алгоритм) вычисления функции.

Указатель функции — средство обращения к алгоритму вычисления функции. Указатель функции имеет вид:

$$f(x_1, x_2, \ldots, x_n)$$

где f — имя функции; x_i $(i = 1, 2, \ldots, n)$ — ее аргументы, которые принято называть фактическими параметрами указателя функции.

В качестве имен функций используются идентификаторы, а в качестве фактических параметров — константы, имена переменных, выражения, а также имена массивов, функций и процедур.

Указатель функции, как и переменная, должен относиться к определенному типу и иметь определенную длину. Тип и длина указателя функции определяются типом и длиной соответствующей функции. В языке предусмотрены функции целого, вещественного, комплексного и логического типов, которые могут иметь как стандартную, так и нестандартную длину.

Примеры указателей функций:

```
P(3.5)
SIN(X)
COS(A—3.14)
F1(X, Y, Z/2)
```

Наличие указателя функции в выражениях указывает на обращение к *подпрограмме* вычисления соответствующей функции. После того как все необходимые вычисления по этой подпрограмме будут выполнены, значение функции присваивается данному указателю функции.

В языке имеется несколько способов задания вычисления функции, которые будут рассмотрены ниже.

Многие часто встречающиеся функции в языке фортран представлены в виде составной части языка и получили название *встроенных функций*. В табл. 1.5 приведены некоторые из этих функций. В этой таблице тип и длина аргумента и функции обозначены следующим образом: целый тип стандартной длины обозначен через I4, вещественный тип стандартной длины — через R4, вещественный тип нестандартной длины — через R8.

Вопросы и упражнения

1. Какое назначение имеет указатель функции в фортран-программе?

2. Какой тип и длину имеет указатель функции Y (X), если функция с именем Y вещественного типа стандартной длины?

3. Чему будет равно значение указателя функции SQRT(X) после выполнения алгоритма вычисления функции $\sqrt{X}$ с именем SQRT, обращение к которому было сделано с помощью указателя функции при $X = 36.0$?

4. Можно ли рассматривать запись Y(X) как указатель функции, если: а) X — имя простой переменной, б) X — имя массива, в) X — имя функции?

5. Можно ли рассматривать запись X(X) в качестве указателя функции?

Таблица 1.5. Встроенные функции

Указатель функции	Функция	Тип и длина	
		аргумента	функции
ALOG(X)	$y = \ln x$	R4	R4
DLOG(X)		R8	R8
ALOG10(X)	$y = \lg x$	R4	R4
DLOG10(X)		R8	R8
EXP(X)	$y = e^x$	R4	R4
DEXP(X)		R8	R8
SQRT(X)	$y = \sqrt{x}$	R4	R4
DSQRT(X)		R8	R8
SIN(X)	$y = \sin x$	R4	R4
DSIN(X)		R8	R8
COS(X)	$y = \cos x$	R4	R4
DCOS(X)		R8	R8
TAN(X)	$y = \operatorname{tg} x$	R4	R4
DTAN(X)		R8	R8
COTAN(X)	$y = \operatorname{ctg} x$	R4	R4
DCOTAN(X)		R8	R8
ARSIN(X)	$y = \arcsin x$	R4	R4
DARSIN(X)		R8	R8
ARCOS(X)	$y = \arccos x$	R4	R4
DARCOS(X)		R8	R8
ATAN(X)	$y = \operatorname{arctg} x$	R4	R4
DATAN(X)		R8	R8
IABS(X)	$y = \lvert x \rvert$	I4	I4
ABS(X)		R4	R4
DABS(X)		R8	R8
MAX0(X1, . . ., XN)	$y = \max(x_1, \ldots, x_n)$	I4	I4
AMAX0(X1, . . ., XN)		I4	R4
MAX1(X1, . . ., XN)		R4	I4
AMAX1(X1, . . ., XN)		R4	R4
MIN0(X1, . . ., XN)	$y = \min(x_1, \ldots, x_n)$	I4	I4
AMIN0(X1, . . ., XN)		I4	R4
MIN1(X1, . . ., XN)		R4	I4
AMIN1(X1, . . ., XN)		R4	R4

6. Указать, какие из приведенных ниже записей можно рассматривать как указатели функций:

1) Y(X,A)	4) X(3.14)	7) 1X(6,A)
2) sin(X)	5) Z(3,14,X)	8) X(Y(2))
3) A(X1,2X2,3)	6) B(1)	9) Y( )

1.8. Арифметические выражения

Арифметические выражения образуются из арифметических операндов, знаков арифметических операций и круглых (открытых и закрытых) скобок.

К арифметическим операндам относятся константы без знака, переменные (имена переменных) и указатели функций целого, вещественного и комплексного типов.

Знаками арифметических операций являются:

** (возведение в степень),
* (умножение),
/ (деление),
+ (сложение),
— (вычитание).

Простейшее арифметическое выражение состоит из одного арифметического операнда. Примерами таких выражений являются:

```
A
3.14
F(X)
```

Здесь F(X) — указатель функции, в котором F — имя функции, X — фактический параметр (аргумент функции).

Более сложные арифметические выражения состоят из нескольких арифметических операндов, соединенных знаками арифметических операций. Примерами таких выражений являются:

```
X + Y
3.5 +7 * K—Z
SQRT(X) ** 2 + C15 * Y(3)
```

Здесь SQRT(X) и Y(3) — указатели функций. Перед первым операндом может стоять знак — (минус), как, например, в следующих выражениях:

```
—SIN(X)
—6+A ** K—Z15
```

Любое из арифметических выражений может быть заключено в круглые скобки и использовано в других арифметических выражениях на правах операнда. Таким образом, арифметическое выражение может иметь весьма сложную структуру с большим количеством вложенных друг в друга скобок.

Примеры арифметических выражений со скобками:

A * (B+C)

1.4/(A+B)—X2

((A+1)*3+2) * X—(X+2) ** 2

При написании арифметических выражений недопустима запись двух или нескольких знаков операций, следующих непосредственно друг за другом. Например, запись A * —B не допустима, тогда как запись A * (—B) есть арифметическое выражение.

Все арифметические операции, за исключением ряда особых случаев, имеют обычный смысл. К особым случаям относятся:

1) в качестве результата деления целой величины на целую берется целая часть частного;

2) в качестве результата степени, основание которой — целая величина, а показатель — целая отрицательная величина, берется целая часть степени;

3) операция возведения в комплексную степень не имеет смысла;

4) если $X < 0$, а Y — величина вещественного типа, то операция X ** Y не имеет смысла;

5) если X — величина комплексного типа, а Y — величина вещественного типа, то операция X ** Y не имеет смысла.

Например, если переменные A и B целого типа и A = 10, B = = 3, то результатом операции A/B будет число 3; или если переменная A целого типа и A = 2, то результатом операции A**(—2) будет число нуль.

При вычислении значения арифметического выражения операции выполняются в строго определенном порядке с учетом скобок и правил старшинства операций. Установлены следующие уровни старшинства операций:

1-й уровень — вычисление указателя функции (вычисляется в первую очередь);

2-й уровень — возведение в степень (вычисляется во вторую очередь);

3-й уровень — умножение и деление (вычисляются в третью очередь);

4-й уровень — сложение и вычитание (вычисляются в четвертую очередь).

Выражение в скобках вычисляется до того, как его значение будет использовано в качестве операнда. Например, выражение B + ((A + B)*C) + A ** 2 будет вычисляться в следующем порядке:

1) A + B (результат обозначим через X)
2) X * C (результат обозначим через Y)
3) B + Y (результат обозначим через W)
4) A ** 2 (результат обозначим через Z)
5) W + Z

Если операции одного и того же уровня старшинства следуют непосредственно друг за другом, то они выполняются в последовательности слева направо. Так, например, выражение A * B/C * D * K будет вычисляться в следующем порядке:

1) A * B (результат обозначим через X)
2) X/C (результат обозначим через Y)
3) Y * D (результат обозначим через Z)
4) Z * K

Однако это правило не распространяется на операции возведения в степень. Операции возведения в степень выполняются в последовательности справа налево. Так, например, выражение A **B ** C **D будет вычисляться в следующем порядке:

1) C ** D (результат обозначим через X)
2) B ** X (результат обозначим через Y)
3) A ** Y

Значение арифметического выражения может относиться к целому, вещественному или комплексному типам и иметь стандартную или нестандартную длину. Правила определения типа и длины значений отдельных арифметических операций в зависимости от типа и длины операндов отражены в табл. 1.6 и 1.7. В этих таблицах приняты следующие обозначения:

$Ц_С$ — целая величина стандартной длины;
$Ц_Н$ — целая величина нестандартной длины;
$В_С$ — вещественная величина стандартной длины;
$В_Н$ — вещественная величина нестандартной длины;
$К_С$ — комплексная величина стандартной длины;
$К_Н$ — комплексная величина нестандартной длины.

Символ × используется для обозначения запрещенной комбинации.

Тип и длина значения арифметического выражения определяются путем применения этих правил ко всем операциям, содержащимся в этом выражении. Порядок применения правил определяется порядком выполнения соответствующих операций.

Таблица 1.6. Тип и длина результата операции AτB, где τ — один из знаков: + (плюс), — (минус), * (умножение), / (деление)

A \ B	$Ц_С$	$Ц_Н$	$В_С$	$В_Н$	$К_С$	$К_Н$
$Ц_С$	$Ц_С$	$Ц_С$	$В_С$	$В_Н$	$К_С$	$К_Н$
$Ц_Н$	$Ц_С$	$Ц_Н$	$В_С$	$В_Н$	$К_С$	$К_Н$
$В_С$	$В_С$	$В_С$	$В_С$	$В_Н$	$К_С$	$К_Н$
$В_Н$	$В_Н$	$В_Н$	$В_Н$	$В_Н$	$К_Н$	$К_Н$
$К_С$	$К_С$	$К_С$	$К_С$	$К_Н$	$К_С$	$К_Н$
$К_Н$	$К_Н$	$К_Н$	$К_Н$	$К_Н$	$К_Н$	$К_Н$

Таблица 1.7. Типы и длины результата операции A**B

A \ B	$Ц_С$	$Ц_Н$	$В_С$	$В_Н$
$Ц_С$	$Ц_С$	$Ц_С$	$В_С$	$В_Н$
$Ц_Н$	$Ц_С$	$Ц_Н$	$В_С$	$В_Н$
$В_С$	$В_С$	$В_С$	$В_С$	$В_Н$
$В_Н$	$В_Н$	$В_Н$	$В_Н$	$В_Н$
$К_С$	$К_С$	$К_С$	×	×
$К_Н$	$К_Н$	$К_Н$	×	×

Так, например, если A и D — переменные вещественного типа стандартной длины; I, L, K — переменные целого типа нестандартной длины, то тип и длина значения арифметического выражения ((A + D) * L + A) ** (I + K) будут определяться следующим образом:

Операция	Результат	Тип результата	Длина результата
A+D	X1	Вещественный	Стандартная
X1*L	X2	Вещественный	Стандартная
X2+A	X3	Вещественный	Стандартная
I+K	X4	Целый	Нестандартная
X3**X4	Значение выражения	Вещественный	Стандартная

Приведение значения целой величины к вещественному или комплексному типу, а также значения вещественной величины к комплексному типу осуществляется по принятым в математике правилам.

Вопросы и упражнения

1. Можно ли утверждать, что арифметическое выражение есть алгебраическое выражение?

2. Можно ли в качестве операнда в арифметическом выражении использовать а) имя массива, б) имя переменной логического типа, в) имя функции вещественного типа?

3. Указать, какие из приведенных ниже записей можно рассматривать как арифметические выражения:

1) 27
2) J(I, X)
3) X+2*a
4) (3.14,2.)
5) 6.30—SIN(X)+2X
6) —((((X))))
7) X+A*(*B)
8) 1.3/B+A(X+2.6)
9) (((((A+1)*3.+2)*3.+3)*3.)+4)*3.+5
10) —0

4. Представить в виде арифметических выражений следующие алгебраические выражения:

1) x^{y^z}

2) $1 + x + \frac{x^2}{2!} + \frac{x^3}{3!}$

3) $1 + \cfrac{x}{1 + \cfrac{x}{1 + x}}$

4) $3 \sin x + 4 \cos^2 x^1 - 1$

5) $(x^y/y^x)^{z\,(y/x)}$

6) $(a + b)\sin a + (a - b)\cos b + \frac{a - b}{\sin a + \cos b}$

7) $\frac{x + 1}{x - 1} + 3{,}6\,\{x - (\sin x + 1)^2 + x^2(\sin x - 1)\}$

8) $-4{,}3 - x^{y(6{,}2-y^6)} + 1$

9) $(a-b) \Big/ \left(c + \dfrac{b}{c + b/(c-d)}\right)$

10) $e^x - 3\sin x + e^{(x^2-1)}$

5. Определить порядок выполнения операций в следующих арифметических выражениях:

1) —A * B/A * A/C * B

2) X ** Y ** Z ** A

3) X + X/(X + Y) * Z/(Z + X) + (X—Y)/Z

4) —3.4 + A * SIN(A + B * 2) ** 3/2—X

5) —X + X ** 2/X—1 + (—6.3 * X—X ** 2 + 4.3)

6. Вычислить значения следующих арифметических выражений:

1) 3/X + X ** 2, если X = 2 и имеет целый тип,

2) 6.98 + A ** (—2), если A = 3 и имеет целый тип,

3) X + 3—X ** 3, если X = —3 и имеет целый тип.

7. Определить тип и длину следующих арифметических выражений:

1) I + F + L, если I, F и L — переменные целого типа нестандартной длины;

2) (M + K * (I+K)) * L—2, если K, I, M — переменные целого типа стандартной длины, а L — переменная целого типа нестандартной длины;

3) ((F * A—I) * F+P) * F+B, если I и F — переменные целого типа стандартной длины, A и B — переменные вещественного типа стандартной длины, P — переменная вещественного типа нестандартной длины;

4) (V/(Z * B—G)+G/Z) * R+1, если B и G — переменные вещественного типа нестандартной длины, V, Z и R — переменные комплексного типа стандартной длины;

5) (L ** F ** F) ** I, если L — переменная целого типа стандартной длины; F и I — переменные целого типа нестандартной длины.

8. Определить тип, длину и значение арифметического выражения

(X ** 2—Y * T+4.1) * Y—6.3 * T/Y+S,

где X — переменная целого типа нестандартной длины, значение которой равно —7, Y и S — переменные вещественного типа стандартной длины, значения которых соответственно равны 4.02 и 105.06E—4, T — переменная комплексного типа стандартной длины, значение которой равно (5.1,0.06).

1.9. Отношения

Отношение имеет вид:

aσb

где *a* и *b* — арифметические выражения целого или вещественного типов, σ — знак операции отношения. Выражения *a* и *b* могут относиться к различным типам и иметь различные длины. В процессе вычисления значения отношения они приводятся к одной длине и к одному типу таким же образом, как и операнды арифметических выражений.

Знаками операций отношений являются следующие последовательности основных символов:

.GT. (больше)
.GE. (больше или равно)
.LT. (меньше)
.LE. (меньше или равно)
.EQ. (равно)
.NE. (неравно)

Результат операции отношения относится к логическому типу. Он принимает значение истина в том случае, если отношение удовлетворяется для входящих в него арифметических выражений и ложь в противном случае.

Так, например, если A, B, C — переменные вещественного типа, M — переменная целого типа, то в качестве отношений могут применяться следующие записи:

```
A.GT.C+B * 3.5
D+G.GT.M ** 2
```

Если, например, A = 374.2, C = 2.6 и B = —3.75; D = —36.1, G = 0.02 и M = 4, то первое из этих отношений принимает значение истина, а второе — ложь.

Вопросы и упражнения

1. Могут ли арифметические выражения комплексного типа входить в состав отношения?

2. Могут ли в состав отношения входить арифметические выражения а) разного типа (целого и вещественного); б) разной длины (стандартной и нестандартной)?

3. Указать, какие из приведенных ниже записей можно рассматривать как отношения:

1) A+B**2<0.6, если A и B — переменные вещественного типа;

2) S*T/2—0.3.NE.2, если S — переменная вещественного типа стандартной длины, T — переменная вещественного типа нестандартной длины;

3) —3.09.GE.SQRT(X)+EXP(X+1), если X — переменная целого типа;

4) X.I.T.A—1, если A — переменная вещественного типа, X — переменная логического типа;

5) Q+S/2.NE.S—Q/2, если S — переменная целого типа нестандартной длины, Q — переменная вещественного типа стандартной длины.

4. Определить значения следующих отношений:

1) A—B/2.I.T.4.7, если A и B — переменные вещественного типа, значения которых соответственно равны —0.6 и 2.3E—2;

2) 0.0.GT.A**(X+Y)—A/X+A/Y, если A — переменная вещественного типа, значение которой равно 6.82, а X и Y — переменные целого типа, значения которых соответственно равны 2 и —3;

3) —X+SIN(Y+X).GE.SQRT(X—Y)+Y, если X и Y — переменные вещественного типа, значения которых соответственно равны 4.5 и —4.5, SIN(Z) — функция sin Z и SQRT(Z) — функция $\sqrt{Z}$.

1.10. Логические выражения

Логические выражения образуются из логических операндов, знаков логических операций и круглых скобок. К логическим операндам относятся константы, переменные, указатели функций логического типа и отношения.

Знаками логических операций являются следующие последовательности символов:

.NOT. (отрицание)
.AND. (логическое умножение)
.OR. (логическое сложение)

Операция отрицания относится к одноместным операциям, а логическое умножение и логическое сложение — к двуместным. Правила выполнения логических операций приведены в табл. 1.8.

Простейшее логическое выражение состоит либо из одного логического операнда, либо из одного логического операнда, перед которым стоит знак операции отрицания. Примерами таких выражений являются:

```
A
.TRUE.
R.GT.C+0.5
.NOT.A
.NOT..TRUE.
.NOT.R.GT.C+0.5
```

где A — переменная логического типа, а R и C — переменные вещественного или целого типа.

Таблица 1.8. Правила выполнения логических операций

A	истина	истина	ложь	ложь
B	истина	ложь	истина	ложь
.NOT.A	ложь	ложь	истина	истина
A.AND.B	истина	ложь	ложь	ложь
A.OR.B	истина	истина	истина	ложь

Более сложные логические выражения состоят из нескольких простейших логических выражений, соединенных знаками двуместных логических операций. Примерами таких выражений являются:

```
A.AND..TRUE.
A.OR.R.GT.C+0.5.AND..NOT.L
```

где A и L — переменные логического типа, а R и C — переменные вещественного или целого типа.

Любое из логических выражений может быть заключено в круглые скобки и использовано в других логических выражениях на правах операнда. Таким образом, логическое выражение может иметь весьма сложную структуру с большим количеством вложенных друг в друга скобок. Примеры логических выражений со скобками:

```
A1.AND.((B1.OR.C1).AND..FALSE.)
Y.OR.((X.OR.Z).AND..NOT.Z.OR.X).AND.Z
```

При вычислении значения логического выражения операции выполняются в последовательности слева направо с учетом правил старшинства операций и скобок. Установлены следующие уровни старшинства операций:

1-й уровень — вычисление арифметических выражений;

2-й уровень — вычисление отношений;
3-й уровень — отрицание;
4-й уровень — логическое умножение;
5-й уровень — логическое сложение.

Выражение в скобках вычисляется до того, как его значение будет использовано в качестве операнда. Например, выражение

```
L.AND..NOT.(M.OR.X.EQ.Y+2).AND.Z
```

будет вычисляться в следующем порядке:

1) Y+2 (результат обозначим через X1),
2) X.EQ.X1 (результат обозначим через X2),
3) M.OR.X2 (результат обозначим через X3),
4) .NOT.X3 (результат обозначим через X4),
5) L.AND.X4 (результат обозначим через X5),
6) X5.AND.Z

Правила определения длины значений отдельных логических операций приведены в табл. 1.9.

Таблица 1.9. Правила определения длины значений логических операций

Длина первого операнда	Операция	Длина второго операнда	Длина результата
—	Отрицание	Стандартная	Стандартная
—	Отрицание	Нестандартная	Нестандартная
Стандартная	Логическое сложение и умножение	Стандартная	Стандартная
Стандартная	Логическое сложение и умножение	Нестандартная	Стандартная
Нестандартная	Логическое сложение и умножение	Стандартная	Стандартная
Нестандартная	Логическое сложение и умножение	Нестандартная	Нестандартная

Длина значения логического выражения определяется путем применения этих правил ко всем операциям, содержащимся в этом выражении. Порядок применения правил определяется порядком выполнения соответствующих операций.

Вопросы и упражнения

1. Какие значения может принимать логическое выражение?

2. Может ли логическое выражение начинаться со знака логической операции?

3. Указать, какие из приведенных ниже записей нельзя рассматривать как логические выражения:

1) A+B2.GE.0.OR.X

2) NOT..TRUE.

3) X.OR.Y.AND.0.6.GT.X+2.7

4) .FALSE..AND..NOT.Y.OR.(Y.OR..NOTY).AND.TRUE.

5) .NOT.((A— C* B+0.23) * A.LT.(B/(C—A)—102.3).AND.A/(B+C—A).OR.X.OR.Y

4. Представить в виде логических выражений следующие высказывания:

1) $(A+B)x-B \leqslant 0.3-A$

2) $x \vee \neg y \wedge (x \vee y) \wedge y$

3) $\neg x \wedge (AB-C)B+d > (A+B)/C \vee x \wedge \neg (x \vee y)$

4) $3.6-2 \neq 2 \wedge x \wedge y \wedge \neg ((x \vee y) \wedge z)$

5) $\neg (a+b-c)/2 < 3.6 \vee (a-b) > c/3$

Примечание. Знаки $\vee$, $\wedge$ и $\neg$ означают соответственно операцию логического сложения, логического умножения и отрицания.

5. Определить порядок выполнения операций в следующих логических выражениях:

1) .NOT.Y.OR.X.AND..NOT.Z

2) A+3.06 * C.EQ.0.OR.X.AND..NOT.Y

3) Y.OR.((X.OR.Z).AND..NOT.Z.OR.X).AND.Z

4) Y.AND.X2J.AND.S.AND..NOT.Y

5) .NOT.(A—D/2) * B.GT.A.AND.X.OR..NOT.Y

6. Определить значения следующих логических выражений:

1) .TRUE.

2) .NOT.X.AND..FALSE. , если переменная X имеет значение .FALSE.

3) .FALSE.OR.A+B.LE.0.6, если переменные A и B имеют соответственно значения: —0.3E02 и 7.3E—2

4) X.OR..NOT.Y.AND..NOT.Z.OR.S—6.3/A+ B.GT.0, если переменные X, Y, Z, S, A и B имеют соответственно значения: .TRUE., .TRUE., .FALSE., —3.4E—1, 6.1, —0.001E4.

Глава 2

ОПЕРАТОРЫ ПРИСВАИВАНИЯ И УПРАВЛЕНИЯ

2.1. Метки

Как указывалось выше, в языке фортран различают два класса операторов: выполняемые и невыполняемые.

Каждый из выполняемых операторов фортран-программы может быть помеченным, т. е. снабжен целой константой без знака, получившей название метки. *Метка* оператора позволяет отличать этот оператор от другого оператора фортран-программы и играет роль адреса этого оператора. Из числа невыполняемых операторов помечаться должны лишь операторы форматов.

Примеры меток:

17
35
005

Метки должны содержать не более пяти цифр и их значения должны лежать в диапазоне от 1 до 99 999.

Метки используются для различных ссылок на операторы, снабженные этими метками. Выполняемые операторы, на которые нет ссылок в других операторах, снабжать метками не обязательно.

Вопросы и упражнения

1. Что собой представляет метка и для каких целей она используется в фортран-программе?

2. Указать, какие из приведенных ниже записей нельзя рассматривать как метки:

1) 0 6) 00002
2) 2040 7) 100000
3) —4 8) 107
4) 2A3 9) 100.
5) 1.1 10) 0000

3. Какие операторы языка фортран могут быть помечены?
4. Обязательно ли помечать операторы фортран-программы?

2.2. Операторы присваивания

К *операторам присваивания* относятся арифметический оператор присваивания, логический оператор присваивания и оператор присваивания меток.

2.2.1. Арифметический оператор присваивания. *Арифметический оператор присваивания* имеет вид:

$$n\ a=b$$

где n — либо пусто (отсутствие каких-либо символов), либо метка, помечающая данный оператор; a — переменная (простая или с индексами) одного из арифметических типов (целого, вещественного, комплексного); b — арифметическое выражение.

Тип и длина левой части a оператора присваивания могут не совпадать соответственно с типом и длиной правой части b.

При выполнении арифметического оператора присваивания выполняются следующие действия:

1) вычисляется значение выражения, содержащегося в правой части оператора;

2) значение правой части оператора приводится к типу и длине левой части с последующим присвоением его переменной, содержащейся в левой части.

Таким образом, можно сказать, что символ = в арифметическом операторе присваивания играет роль знака операции присваивания.

Приведение значения правой части к типу левой части осуществляется в соответствии со следующими соглашениями:

1. Пусть переменная a — целого типа, выражение b — вещественного типа. Тогда в качестве значения переменной a берется целая часть значения выражения b.

2. Пусть переменная a — целого типа, выражение b — комплексного типа. Тогда в качестве значения переменной a берется целая часть действительной компоненты значения выражения b.

3. Пусть переменная a — вещественного типа, выражение b — целого типа. Тогда в качестве значения переменной берется значение выражения b.

4. Пусть переменная a — вещественного типа, выражение b — комплексного типа. Тогда в качестве значения переменной a берется действительная компонента значения выражения b.

5. Пусть переменная a — комплексного типа, выражение b — целого или вещественного типа. Тогда в качестве значения действительной компоненты переменной a берется значение выражения b, а в качестве коэффициента при мнимой части переменной a — число нуль.

Примеры арифметических операторов присваивания:

```
T =4.6
5  X=Q+PS/R
03 Y=(X+R2)*T—R2
```

Вопросы и упражнения

1. Какую роль играет знак равенства (символ =) в арифметическом операторе присваивания?

2. Назвать последовательность действий при выполнении арифметического оператора присваивания.

3. Каким образом выполняется преобразование значения арифметического выражения, входящего в состав арифметического оператора присваивания, к типу и длине переменной, входящей в состав левой части этого оператора?

4. Написать арифметические операторы присваивания для вычисления значений величины по следующим формулам:

1) $y=\sin^2 a+\cos(a-3.14)+1$

2) $$x=\cfrac{y}{1+\cfrac{y^2}{1+\cfrac{3y^3}{1+\cfrac{y}{1+y}}}}$$

3) $P(x)=((((a_5x+a_4)x+a_3)x+a_2)x+a_1)x+a_0$

2.2.2. Логический оператор присваивания. *Логический оператор присваивания имеет вид:*

```
n a=b
```

где n — либо пусто, либо метка, помечающая данный оператор; a — переменная (простая или с индексами) логического типа; b — логическое выражение.

При выполнении такого оператора выполняются следующие действия:

1) вычисляется значение логического выражения, содержащегося в правой части оператора;

2) вычисленное значение выражения приводится к длине левой части с последующим присвоением его переменной, содержащейся в левой части.

Как и в арифметическом операторе присваивания, в логическом операторе присваивания символ = играет роль знака операции присваивания.

Примеры логических операторов присваивания:

```
  A=.TRUE.
1 B=X*B—C.GE.1.0
5 Z=X.OR.Y.EQ.5
```

Вопросы и упражнения

1. Может ли логический оператор присваивания быть непомеченным?

2. Назвать последовательность действий при выполнении логического оператора присваивания.

3. Составить операторы присваивания величине X значений следующих высказываний:

1) $a \neq b$

2) $(a+b)/a+2.3 \geqslant 0.3$

3) $\frac{a}{b+c} - 2b = 2c \wedge a \geqslant c$

4) $\neg(x \vee y) \wedge x \vee \neg y$

5) $x^2 - a > 0.2 \vee e^x < x+2 \wedge \sqrt{x-1} \leqslant 0$

4. Определить значение переменной X при выполнении приведенных ниже логических операторов присваивания:

1) X=A.GT.B, если переменные A и B имеют соответственно значения 3.6 и —4.7;

2) X=X.OR.A+B.NE.6.02, если переменная X имеет значение .TRUE., а переменные A и B имеют соответственно значения: —0.4E03 и 12.6.

2.2.3. Оператор присваивания меток. *Оператор присваивания меток* имеет вид:

n ASSIGN *m* TO *k*

где *n* — либо пусто, либо метка, помечающая данный оператор; *m* — целая константа без знака, совпадающая, как правило, с меткой некоторого оператора фортран-программы; *k* — простая переменная целого типа стандартной длины.

Оператор присваивания меток относится к числу выполняемых операторов. При выполнении этого оператора переменная *k* принимает значение метки *m*. Именно поэтому оператор и называется оператором присваивания метки. Так, например, в результате вы-

полнения оператора

```
ASSIGN 15 TO L
```

переменной L будет присвоено значение метки 15.

Переменная, получившая значение метки, используется в операторе перехода по предписанию (см. п. 2.3.2).

Отметим особенности таких переменных:

1) переменная, получившая значение метки, не может использоваться как переменная, имеющая численное значение до тех пор, пока она его не получит с помощью арифметического оператора присваивания или оператора ввода;

2) переменная, получившая с помощью оператора присваивания или оператора ввода численное значение, совпадающее с меткой, не может использоваться в операторе перехода по предписанию до тех пор, пока она не получит значение этой метки с помощью оператора присваивания метки.

Вопросы и упражнения

1. Может ли в операторе

n ASSIGN *m* TO *k*

m совпадать с *n*? Обязательно ли *m* должна совпадать с меткой некоторого оператора фортран-программы?

2. Может ли переменная *k*, которой при выполнении оператора ASSIGN *m* TO *k* присваивается метка *m*, быть нестандартной длины?

3. Допустима ли последовательность операторов

```
ASSIGN 2 TO X
A=X**3+X—2.3
```

в фортран-программе?

4. Указать, какие из приведенных ниже операторов присваивания метки не содержат ошибок:

```
1)  ASSIGN A TO B
2) 2 ASSIGN 0006 TO J
3)  ASSIGN TO
4)  ASSIGN X TO 2
5)  ASSIGN 3 TO Y
```

2.3. Операторы перехода

Обычно операторы программы выполняются последовательно друг за другом. Это означает, что после выполнения первого оператора будет выполняться второй, непосредственно следующий за

первым, и т. д. В последующих разделах рассматриваются *операторы перехода*, которые относятся к числу выполняемых операторов, эти операторы позволяют изменять порядок выполнения операторов.

К операторам перехода относятся: оператор безусловного перехода, оператор перехода по предписанию и оператор перехода по вычислению.

2.3.1. Оператор безусловного перехода. *Оператор безусловного перехода* имеет вид:

n GO TO *m*

где *n* — либо пусто, либо метка, помечающая данный оператор; *m* — метка некоторого другого оператора.

Действие оператора безусловного перехода заключается в передаче управления оператору, помеченному меткой *m*. Так, например, после выполнения оператора 5 GO TO 3 будет выполняться оператор, помеченный меткой 3.

Вопросы и упражнения

1. Какому из операторов фортран-программы будет передано управление в результате выполнения оператора

GO TO 16

2. Следует ли рассматривать оператор перехода GO TO 4 как ошибочно написанный, если в фортран-программе нет оператора с меткой 4?

3. Указать, какие из приведенных ниже операторов перехода содержат ошибки:

1) GO TO X
2) 3 GO TO 0001
3) 4 GO TO 4
4) GO TO 21
5) GO TO

4. Что произойдет в результате выполнения оператора

10 GO TO 10

2.3.2. Оператор перехода по предписанию. *Оператор перехода по предписанию* имеет вид:

n GO TO *i*, (n_1, n_2,. . ., n_k)

где *n* — либо пусто, либо метка, помечающая данный оператор; *i* — простая переменная целого типа стандартной длины; n_j (j = = 1, 2,. . ., k) — метки некоторых операторов.

К моменту выполнения этого оператора переменной *i* должно быть присвоено значение метки с помощью оператора присваивания

меток, которое должно совпадать с одной из меток n_j. В этом случае оператор перехода по предписанию обеспечивает передачу управления оператору, метка которого совпадает со значением переменной i. Если какое-либо из перечисленных выше условий не выполняется, то действие оператора перехода по предписанию не определено.

Приведем несколько примеров:

1. Пусть задана последовательность операторов

```
ASSING 7 TO K
GO TO K, (12,7,15,3)
```

В результате их выполнения управление будет передано оператору, помеченному меткой 7.

2. Пусть задана последовательность операторов

```
ASSIGN 7 TO K
ASSIGN 15 TO K
GO TO K, (12,7,15,3)
```

В результате их выполнения управление будет передано оператору, помеченному меткой 15.

Вопросы и упражнения

1. Может ли в операторе

```
GO TO K, (2,003,10,6)
```

переменная K быть а) вещественного типа, б) нестандартной длины?

2. Может ли последовательность констант $n_1, n_2, \ldots, n_k$ оператора перехода по предписанию

```
GO TO i, (n1, n2,. . ., nk)
```

содержать константы, которые не используются в качестве меток в фортран-программе, в состав которой входит рассматриваемый оператор?

3. Что произойдет в результате выполнения приведенных ниже операторов

```
   ASSIGN 10 TO J
10 GO TO J, (6,3,4,10,2)
```

4. Указать, какие из приведенных ниже операторов не содержат ошибок:

```
1)   GO TO 26
2)   GO TO L, (23,0,4,10)
```

```
3)     GO TO M
4)     GO TO 4A,(7,14,37)
5) 20  GO TO N, (1,6,01,4)
```

5. Указать метку оператора, которому будет передано управление в результате выполнения следующих групп операторов:

```
1)    ASSIGN 26 TO I
    6 GO TO I,(8,26,3,2)
2)    GO TO 12
3)    GO TO 3
    4 ASSIGN 6 TO X
      GO TO X, (105,3,6)
    3 GO TO 4
```

2.3.3. Оператор перехода по вычислению. *Оператор перехода по вычислению* имеет вид:

$$n \text{ GO TO } (n_1, n_2, \ldots, n_k), t$$

где n — либо пусто, либо метка, помечающая данный оператор; n_j $(j = 1, 2, \ldots, k)$ — метки некоторых операторов; t — простая переменная целого типа стандартной или нестандартной длины.

Этот оператор передает управление либо оператору, помеченному меткой n_i (если $1 \leqslant i \leqslant k$), либо следующему за ним оператору, или $i > k$ или $i = 0$.

К моменту выполнения оператора перехода по вычислению переменной t должно быть присвоено значение. В частности, это можно выполнить с помощью арифметического оператора присваивания.

Рассмотрим примеры.

1. В результате выполнения операторов

```
X = 3
GO TO (3,5,17,40),X
```

где X — переменная целого типа, управление будет передано оператору, помеченному меткой 17.

2. Пусть N — переменная целого типа и N=5. Тогда оператор перехода GO TO (7,13,8),N передаст управление оператору, следующему за данным оператором перехода.

Вопросы и упражнения

1. Может ли значение переменной t, указанной в операторе

GO TO $(n_1, n_2, \ldots, n_k)$, t,

использоваться в других операторах фортран-программы, содержа-

щей указанный оператор, например, в арифметическом операторе присваивания?

2. Могут ли в операторе GO TO (n_1, n_2, . . ., n_k), i быть одинаковые константы в списке n_1, n_2, . . ., n_k?

3. Пусть заданы операторы

```
L=12
GO TO (7,4,4,4,12,12),L

X=L+0.6
```

где L — переменная целого типа. Какой из операторов получит управление после выполнения указанного оператора перехода по вычислению?

4. Указать, какие из приведенных ниже операторов не содержат ошибок:

1) GO TO (13,4,2,8),J
2) GO TO X, (6,2,10)
3) GO TO (9,10,10,10) K
4) GO TO (3,4,5,6,7,8)
5) GO TO (105,700,3,1), X, если X — переменная вещественного типа.

5. Указать метку оператора, которому будет передано управление в результате выполнения следующих групп операторов:

1) 2 GO TO (4,1,1,2,7),J, если J = 3;
2) ASSIGN 5 TO K
   GO TO K,(2,1,6,5,10)
3) GO TO (8,9,10,12,8),L, если L = 11;
4) GO TO 15

2.4. Условные операторы

К условным операторам относятся условный арифметический оператор и условный логический оператор. *Условные операторы* относятся к числу выполняемых операторов.

2.4.1. Условный арифметический оператор. *Условный арифметический оператор* имеет вид:

$$n \text{ IF } (e)\ n_1,\ n_2,\ n_3$$

где n — либо пусто, либо метка, помечающая данный оператор; e — арифметическое выражение целого или вещественного типа; $n_i (i = 1, 2, 3)$ — метки некоторых других операторов.

При выполнении условного арифметического оператора вычисляется значение арифметического выражения и передается управле-

вие оператору, помеченному меткой n_1, n_2 или n_3, если значение арифметического выражения e соответственно меньше нуля, равно нулю или больше нуля.

Приведем несколько примеров:

1) Пусть A и B — переменные целого типа, причем A = 5, B = 7. Тогда оператор IF (A — B)3,7,4 передаст управление оператору, помеченному меткой 3.

2) Пусть A — переменная вещественного типа и A = 3.5. Тогда оператор IF (A + 0.5) 15,9,14 передаст управление оператору, помеченному меткой 14.

3) Пусть A и B — переменные целого типа, и пусть A = —8, B = 8. Тогда оператор IF (A + B) 1,9,2 передаст управление оператору, помеченному меткой 9.

Вопросы и упражнения

1. Может ли арифметическое выражение, входящее в состав условного арифметического оператора, относиться к комплексному типу?

2. Перечислить действия, которые выполняются при выполнении условного арифметического оператора.

3. Указать, какие из приведенных ниже операторов содержат ошибки:

```
1) 2 IF (—6.3)6,10,4
2)   IF (A*B—0.4E—2)9,0,12
3)   IF (A—(0.4,—0.3))1,3,4
```

если A и B — переменные вещественного типа,

```
4) IF (.TRUE.) 1,4,3
5) IF (A) L,2,1
```

если A и L — переменные целого типа.

4. Указать метку оператора, которому будет передано управление в результате выполнения следующих групп операторов:

```
1) IF (.27D10)6,4,1
2) IF (3*A+0.3) 2,5,9, если A = —0.4
3) B = —4.1
   S = 3*B—6.1
   IF (1—S) 8,12,14
```

2.4.2. Условный логический оператор. *Условный логический оператор* имеет вид:

n IF (*e*) *s*

где *n* — либо пусто, либо метка, помечающая данный оператор; *e* — логическое выражение; *s* — выполняемый оператор, отличный от

оператора цикла, о котором будет сказано несколько позже, и от условного логического оператора.

При выполнении условного логического оператора происходит вычисление значения логического выражения *e* и передача управления либо оператору *s* (если *e* есть **истина**), либо оператору, следующему за данным условным логическим оператором (если *e* есть **ложь**).

Приведем несколько примеров:

1. Пусть логическая переменная L имеет значение **ложь**. Тогда оператор 5 IF (.NOT.L) GO TO 7 передаст управление оператору, помеченному меткой 7.

2. Пусть логические переменные L, B и C имеют значения соответственно **ложь**, **ложь** и **истина**. Тогда в результате выполнения оператора IF (L) B=C значение B не изменится, так как оператор B = C выполняться не будет, а управление будет передано следующему оператору.

3. Пусть логическая переменная L имеет значение **истина**, а переменные целого типа A и B — соответственно 2 и 7. Тогда в результате выполнения оператора IF (L) A = B*3 переменная A получит значение 21, после чего управление будет передано следующему оператору.

Вопросы и упражнения

1. Перечислить действия, которые выполняются при выполнении условного логического оператора.

2. Указать, какие из приведенных ниже операторов не содержат ошибок:

```
1) 4 IF (.FALSE.) A=B
2)   IF (A.GT.B) IF (.TRUE.) C=D
3)   IF (.TRUE.) IF (A)3,4,7
4)   IF (X.NE.0) GO TO 6
```

3. Определить, какой из операторов получит управление после выполнения следующих групп операторов:

```
1)   IF (X+B*Y.GT.0) IF (X+2.07)3,5,4
     4 A=(B+X)*Y
```

если X = —2.07, B=1.4, Y =4.25

```
2) IF (X.OR.Y.AND.Z) IF (A—B) 076,24,31
     IF (A+B) 6,104,27
```

если X=.TRUE., Z=Y=.FALSE., A=3.6, B=4.1

```
3) IF (X.AND..NOT.Y) GO TO 104
     X=A+B.GE.0.0
```

если X=.TRUE., Y=.FALSE.

4. Написать операторы, обеспечивающие переход:

1) к оператору с меткой 6, если $A > 3$, и к оператору с меткой 30 в противном случае;

2) к оператору с меткой 2, если арифметическое выражение $A+B/C$ принимает значение больше нуля, к оператору с меткой 4, если это выражение равно нулю, и к оператору с меткой 9, если это выражение принимает значение меньше нуля;

3) к оператору с меткой 100, если точка x принадлежит одновременно отрезкам $[a, b]$ и $[c, d]$; к оператору с меткой 104, если x принадлежит отрезку $[a, b]$, но не принадлежит отрезку $[c, d]$.

5. Составить последовательность операторов для вычисления величины $x = 2a + b - 1$, если $a \geqslant b$, и $y = -x + 0{,}64 \cdot 10^2$ в противном случае.

6. Составить последовательность операторов для вычисления величины $x = f(y) - 26.3$, где $y = z + 2$

$$f(y) = \begin{cases} y^3 - 0{,}3, & \text{если } y < 0 \\ 0, & \text{если } 0 \leqslant y \leqslant 1 \\ y^2 + y, & \text{если } y > 1 \end{cases}$$

2.5. Вспомогательные операторы управления

К этой группе операторов относятся операторы продолжения, паузы, возврата и останова. Все эти операторы относятся к числу выполняемых.

2.5.1. Оператор продолжения. *Оператор продолжения* имеет вид:

n CONTINUE

где *n* — либо пусто, либо метка, помечающая данный оператор.

Этот оператор не выполняет никаких действий и обычно используется в качестве конечного оператора цикла (см. п. 2.6)

2.5.2. Оператор паузы. *Оператор паузы* имеет вид:

n PAUSE *m*

где *n* — либо пусто, либо метка, помечающая данный оператор; *m* — либо пусто, либо целая константа без знака, содержащая от одной до пяти цифр, либо символьная константа без указателя длины.

Этот оператор останавливает выполнение фортран-программы и выдает на пульт управления следующий текст:

PAUSE *m*

Если *m* — пусто, то вместо *m* выдается последовательность из пяти нулей. Так, например, при выполнении оператора PAUSE будет выдан текст PAUSE 00000, при выполнении оператора PAUSE 5 будет выдан текст PAUSE 5, а при выполнении оператора PAUSE 'ВВОД ОКОНЧЕН' будет выдан текст PAUSE 'ВВОД ОКОНЧЕН'.

После останова программа может быть продолжена со следующего оператора по специальному указанию с пульта управления.

2.5.3. Оператор возврата. *Оператор возврата* имеет вид:

n RETURN

где *n* — либо пусто, либо метка, помечающая данный оператор.

Оператор возврата обеспечивает передачу управления (возврат) в ту программную единицу, из которой было передано управление данной программной единице. Если данная программная единица является подпрограммой, к которой было обращение из основной программы, то оператор RETURN обеспечивает возврат в основную программу. Если же данная программа является основной программой, которая получила управление от управляющей программы операционной системы, то оператор RETURN обеспечит возврат в управляющую программу операционной системы.

2.5.4. Оператор останова. *Оператор останова* имеет вид:

n STOP *m*

где *n* — либо пусто, либо метка, помечающая данный оператор; *m* — либо пусто, либо целое число без знака, содержащее от одной до пяти цифр.

Оператор останова используется только в основной программе и обеспечивает завершение выполнения основной программы. Если *m* не пусто, то в процессе выполнения оператора останова на пульт управления выдается текст STOP *m*.

Вопросы и упражнения

1. Какие действия выполняются оператором продолжения?
2. Какие действия выполняются оператором паузы?
3. Можно ли после останова по оператору паузы продолжить выполнение фортран-программы?
4. Что будет выдано на пульт управления в результате выполнения оператора PAUSE 'STOP'?
5. Пусть подпрограмма с именем SUM получила управление от подпрограммы с именем Q, которая, в свою очередь, получила управление от подпрограммы с именем ROS. Указать, какая из названных подпрограмм получит управление в результате выполне-

ния оператора возврата, входящего в состав подпрограммы с именем SUM.

6. Какие действия выполняются оператором останова?

7. Может ли оператор останова входить в состав подпрограмм?

8. Указать, какие из приведенных ниже операторов содержат ошибки:

```
 1)      CONTINUE
 2)    4 CONTINUE
 3)      PAUSE
 4)  105 PAUSE 0026001
 5)      PAUSE 3HSUM
 6)    2 PAUSE 'SUM'
 7)      RETURN.
 8)   10 RETURN
 9)      STOP
10)      PAUSE 'STOP'
11)      STOP 'F1'
12)      STOP 34
13)      PAUSE —3741
```

9. Какой из операторов обеспечивает окончательное завершение основной программы?

2.6. Оператор цикла

Оператор цикла имеет вид:

$$n \text{ DO } m \; i = m_1, m_2, m_3$$

где n — либо пусто, либо метка, помечающая данный оператор; m — метка другого оператора; i — *параметр цикла*; m_1 — *начальное значение параметра цикла*; m_2 — *конечное значение параметра цикла*; m_3 — *приращение параметра цикла*. Конструкцию $i = m_1, m_2, m_3$ принято называть *заголовком цикла*.

В качестве метки m указывается метка выполняемого оператора. Этот оператор должен размещаться в программе после оператора цикла; он называется конечным *оператором цикла*. Оператор, расположенный непосредственно за оператором цикла, называется *начальным оператором цикла*. Начальный оператор цикла может совпадать с конечным. Например, первый из операторов

```
    DO 5 L=1,9
5   X=X+Y(L)
```

есть оператор цикла, второй — конечный оператор цикла. Конеч-

ный оператор цикла есть в то же время и начальный оператор цикла.

В качестве параметра цикла может использоваться только простая переменная целого типа, а в качестве начального значения, конечного значения и приращения параметра цикла — либо целые константы без знака, не равные нулю, либо простые переменные целого типа. К моменту выполнения оператора цикла значения этих переменных должны быть больше нуля.

Приведем примеры операторов цикла:

1. Пусть X — переменная целого типа. Тогда конструкции вида:

а) DO 10 X=1,20,3

б) DO 6 X=15,4,1

есть операторы циклов.

2. Пусть I, J, K — переменные целого типа. Тогда конструкция

DO 2 K=1,I,J

есть оператор цикла. Переменные I и J могут принимать значения больше нуля.

Если m_3 — величина постоянная и равна единице, то в заголовке цикла ее можно не указывать. В этом случае заголовок цикла будет иметь вид:

$$i=m_1, m_2$$

Например, оператор цикла DO 3 K=2,17,1 можно представить в виде DO 3 K=2,17.

Группу операторов, которая начинается с оператора цикла и заканчивается конечным оператором цикла, будем называть *циклом*, а группу операторов, которая начинается с начального оператора цикла и заканчивается конечным оператором цикла, будем называть *областью оператора цикла*. Так в приведенном выше примере оба оператора образуют цикл, а второй оператор образует область цикла.

Приведем другой пример цикла

```
   DO 2 J=1,20,2
   IF (A(J))2,3,3
3  S=S+A(J)
2  CONTINUE
```

Здесь первый оператор является оператором цикла, второй — начальным оператором цикла, последний оператор — конечным оператором цикла. Цикл образует все четыре оператора, а область цикла образуют второй, третий и четвертый операторы цикла.

Оператор цикла определяет границы цикла, значения параметра цикла и количество выполнений операторов, входящих в область цикла, заставляя их повторно выполняться. Это осуществляется следующим образом: величине i присваивается значение m_1 и управление передается начальному оператору цикла. После выполнения конечного оператора цикла к величине i прибавляется величина m_3. Если после этого значение i еще не стало больше m_2, то управление снова передается начальному оператору цикла. Далее процесс повторяется. Если в процессе проверки оказалось, что значение i больше m_2, то управление передается оператору, следующему за конечным оператором цикла.

Совершенно очевидно, что всякий цикл можно заменить эквивалентным «циклом», не организованным без использования оператора цикла. Действительно пусть n_1 — метка оператора цикла, n_2 — метка начального оператора цикла, n_3 — метка конечного оператора цикла. Тогда действие цикла

n_1 DO n_3 K =M,N,H

n_2 ⟨начальный оператор цикла⟩

. . . . . .

n_3 ⟨конечный оператор цикла⟩

можно описать с помощью других операторов языка следующим образом:

n_1 K=M

n_2 ⟨начальный оператор цикла⟩

. . . . . .

n_3 ⟨конечный оператор цикла⟩

K=K+H
IF (K.LE.N) GO TO n_2

Если $N \geqslant M$, то число выполнений области оператора цикла определяется выражением

$$\left[\frac{N-M}{H}\right]+1$$

где квадратные скобки означают выделение целой части. Если же $N<M$, то область оператора цикла выполняется один раз.

Рассмотрим примеры.

1. Цикл

```
    DO 2 N=1,12,2
2   X=X+Y(N)
```

можно организовать без оператора цикла следующим образом:

```
      N=1
2   X=X+Y(N)
      N=N+2
      IF (N.LE.12) GO TO 2
```

Количество выполнений области данного цикла равно:

$$\left[\frac{12-1}{2}\right]+1=6$$

2. Цикл

```
    DO 1 K=10,8
1   P=P+2**K
```

будет выполняться один раз при значении параметра цикла K, равном 10.

При организации циклов необходимо учитывать следующие дополнительные условия:

1. Конечный оператор цикла не может быть оператором цикла, оператором перехода, условным оператором, оператором возврата, паузы и останова. Если цикл по условию задачи должен быть закончен одним из «запрещенных» операторов, то в качестве конечного оператора цикла рекомендуется использовать оператор продолжения, разместив его после «запрещенного» оператора. Так, например, можно написать такой цикл:

```
      DO 2 K=1,150
      IF (X(K))3,1,1
1   Y(K)=X(K)**3
3   IF ((Y(K))2,2,4
2   CONTINUE
```

2. Величины, входящие в заголовок цикла, не должны изменять свои значения внутри области оператора цикла. Например, цикл

```
      DO 3 L=1,10
      X=X+Y(L)
3   L=L*2+0.5
```

содержит ошибку, так как один из операторов (конечный оператор) изменяет значение параметра цикла L. Точно так же цикл

```
    DO 1 L=1,10,H
    H=H+1
1   X=X+Y (L)
```

содержит ошибку, так как начальный оператор цикла изменяет приращение H параметра цикла.

3. Область некоторого (внешнего) оператора цикла может содержать другие (внутренние) операторы циклов. При этом область любого *внутреннего оператора цикла* должна быть подмножеством области *внешнего оператора цикла*. Например, приведенные ниже циклы вложены друг в друга. И это допустимо.

```
    DO 1 I=1,10
    DO 1 J=1,3
1   S=S+A (I,J)
```

Здесь внутренний цикл составляют второй и третий операторы, внешний — первый, второй и третий. Однако недопустима следующая организация циклов:

```
    DO 31 I=1,20
    DO 32 K=1,20
31  S=S+A(I,K)
32  B(K)=S
```

когда один цикл пересекается с другим.

4. Вход в область оператора цикла, вообще говоря, разрешается только через оператор цикла. Однако допускается вход в область самого внутреннего оператора цикла, минуя оператор цикла, но только в том случае, если ему предшествовал (по времени) выход из этой области (еще до окончания цикла) и была обеспечена сохранность значений величин, входящих в заголовок цикла. Например, в приведенных ниже операторах

```
    S=0
    DO 1 J=1,25
1   S=S+X(J)
```

вход в область цикла организован правильно от первого оператора к оператору цикла. В приведенных ниже операторах правильно организован вход к внутреннему оператору цикла из вне этого цикла с помощью оператора GO TO 2:

```
      S=0
      DO 2 J=1,10
      IF(X(J))1,2,2
    2 S=S+X(J)
      GO TO 3
    1 X(J)=X(J)*2+0.6
      GO TO 2
    3 S=S+2
```

так как этому входу в цикл предшествовал выход из цикла и выполнение действий, которые не меняют значения параметра цикла J.

5. Область оператора цикла может содержать указатели функций, посредством которых осуществляются обращения к подпрограммам вычисления функций, а также операторы процедур, посредством которых осуществляются обращения к подпрограммам — процедурам. Например, в приведенном ниже цикле

```
      DO 1 K=1,50,2
    1 S=S+SQRT(A(K))
```

имеется обращение к подпрограмме вычисления функции $\sqrt{A(K)}$.

6. Один и тот же оператор может являться конечным оператором нескольких вложенных друг в друга циклов.

Действие цикла:

```
DO n K1=M1,N1,H1
m1 ⟨начальный оператор 1-го цикла⟩
. . . . . .
DO n K2=M2,N2,H2
m2 ⟨начальный оператор 2-го цикла⟩
. . . . . .
. . . . . .
DO n KL=ML,NL,HL
ml ⟨начальный оператор l-го цикла⟩
. . . . . .
n ⟨конечный оператор⟩
```

в котором оператор с меткой n является конечным для нескольких вложенных друг в друга циклов, можно описать с помощью других операторов языка без использования операторов цикла следующим образом:

```
      K1=M1
```

```
      m1 (начальный оператор 1-го цикла)
      . . . . . .
      K2=M2
      m2 (начальный оператор 2-го цикла)
      . . . . . .
      KL=ML
      ml (начальный оператор l-го цикла)
      . . . . . .
      n (конечный оператор)
      KL=KL+HL
      IF (KL.LE.NL) GO TO ml
      . . . . . .
      K2=K2+H2
      IF (K2.LE.N2) GO TO m2
      K1=K1+H1
      IF(K1.LE.N1) GO TO m1
```

Так вложенные друг в друга циклы

```
      DO 1I=1,10
      DO 1J=1,3
    1 S=S+A(I,J)
```

можно организовать без использования операторов цикла следующим образом:

```
        I=1
    2   J=1
    1   S=S+A(I,J)
        J=J+1
        IF(J.LE.3) GO TO 1
        I=I+1
        IF (I.LE.10) GO TO 2
```

7. Если некоторый оператор является конечным оператором нескольких вложенных друг в друга циклов, то ему можно передавать управление (с помощью операторов перехода, условных операторов и т. д.) только из области самого внутреннего оператора цикла.

8. После выхода из цикла по условию окончания цикла, определяемому заголовком, значение параметра цикла становится неопределенным.

Вопросы и упражнения

1. Что называется циклом?

2. Какие операторы называются начальным и конечным операторами цикла?

3. Что называется областью цикла?

4. Может ли параметр цикла быть а) переменной с индексом, б) простой переменной вещественного типа?

5. Может ли параметр цикла принимать нулевое и отрицательные значения?

6. В каком случае можно опустить значение приращения параметра цикла в заголовке цикла?

7. Что определяет оператор цикла?

8. Какие операторы не могут быть конечными операторами цикла?

9. Указать, какие из приведенных операторов цикла не содержат ошибок:

1) DO 4 I=10,2,2 если I — переменная целого типа;

2) DO 1 J=0,10 если J — переменная целого типа;

3) 4 DO 2 K=1,L,2 если K и L — переменные целого типа и L=0 перед выполнением оператора цикла;

4) 3 DO 1 X(I)=2,10,3 если массив X относится к целому типу.

10. Указать, какие из приведенных ниже циклов содержат ошибки:

```
1)    DO 1 K=2,10,3
      A(K)=A(K-1)+1
    1 IF (A(K))2,2,3
    2 S=S+A(K)
    3 PAUSE

2)    DO 6 I=1,10
      A(I)=A(I)*2
      DO 7 J=1,4
    6 B(J)=A(I)*B(J)-1
    7 S=S+B(J)

3)    DO 2 K=1,3
      DO 2 L=1,10,2
    2 S=S*A(K,L)
```

11. Определить количество выполнений следующих циклов:

```
1)    DO 1 K=1,50,3
    1 S=S+A(K)
```

```
2)   L=2
     N=10
     DO 2 J=L,N
   2 S=S*A(J)
```

12. Написать последовательность операторов, содержащую цикл, для вычисления значения функции $y_j = x_j + \sin x_j$ для всех $0 \leqslant x_j \leqslant 1$, если $x_j = x_{j-1} + 0.2$ и $x_0 = 0$.

13. Написать последовательность операторов, содержащую цикл, для вычисления величины

$$a = \sum_{i=1}^{40} b_i,$$

где b_i — компоненты вектора b (b_1, b_2,..., b_{40}).

14. Написать последовательность операторов, содержащую циклы, для вычисления величины

$$B = \sum_{i=1}^{10} \sum_{j=1}^{15} a_{ij},$$

где a_{ij} — элементы матрицы $a = (a_{ij})$ ($i = 1, 2, \ldots, 10$; $j = 1, 2, \ldots, 15$).

Глава 3

ОПЕРАТОРЫ СПЕЦИФИКАЦИЙ. ОСНОВНАЯ ПРОГРАММА

Свойства величин и массивов, используемых в фортран-программе, должны быть описаны в этой программе. Такими свойствами величин являются тип (целый, вещественный, комплексный, логический) и длина (стандартная и нестандартная). Для массивов, кроме того, должна быть описана размерность (число измерений и максимальное значение каждого измерения). Указанные свойства величин и массивов можно описать с помощью *операторов описания типов и операторов размеров, которые относятся к числу невыполняемых операторов.*

Значения величин, относящиеся к различным модулям, могут быть совмещены в оперативной памяти. Указание на совмещение значений величин в памяти можно задать с помощью *операторов общих областей*, а указание на некоторую экономию памяти внутри одного модуля можно осуществить с помощью *операторов эквивалентности.*

3.1. Операторы описания типа

К операторам описания типа относятся явный оператор типа и неявный оператор типа.

3.1.1. Явный оператор типа. *Явный оператор типа* имеет вид:

$$t*s_1 a_1(k_1)/x_1/, \ a_2 s_2(k_2)/x_2/, \ \ldots, \ a_n s_n(k_n)/x_n/$$

где t — *указатель типа*; s — общий *указатель длины*; a_i ($i = 1, 2, \ldots, n$) — имя величины (переменной, массива или функции), относящейся к типу t; s_i ($i = 1, 2, \ldots, n$) — указатель длины величины a_i; k_i ($i = 1, 2, \ldots, n$) — *указатель размерности* величины a_i; x_i ($i = 1, 2, \ldots, n$) — начальное значение величины a_i. Параметры s, s_i, (k_i) и $/x_i/$ могут отсутствовать.

В качестве указателя типа используются следующие служебные слова:

1) INTEGER — для указания целого типа;

2) REAL — для указания вещественного типа;

3) COMPLEX — для указания комплексного типа;

4) LOGICAL — для указания логического типа.

В качестве указателя длины используются:

1) *4 — для указания стандартной длины величин целого и вещественного типов;

2) *8 — для указания нестандартной длины величин вещественного типа и стандартной длины величин комплексного типа;

3) *2 — для указания нестандартной длины величин целого типа;

4) *16 — для указания нестандартной длины величин комплексного типа;

5) *1 — для указания нестандартной длины величин логического типа.

Таким образом, указатель длины представляет собой символ *, за которым следует целое число без знака, указывающее, сколько байтов памяти отводится под значение данной величины.

Длина величины a_i устанавливается по указателю s_i. Если указатель s_i отсутствует, то длина величины a_i устанавливается по указателю s. Если же указатель s отсутствует, то длина величины a_i считается стандартной.

Указатели размерности задаются только для массивов и имеют вид:

$$(T_1, T_2, \ldots, T_m)$$

где m — размерность массива ($m \leqslant 7$); T_j ($j = 1,2,\ldots, m$) — максимальное значение j-го индекса массива. В качестве максимальных значений индексов указываются либо целые константы без знаков, либо простые переменные целого типа стандартной длины. Переменные в качестве максимальных размеров индексов могут использоваться только в подпрограммах-функциях и подпрограммах-процедурах и только в том случае, если соответствующий массив является формальным параметром.

Начальные значения могут быть присвоены только переменным и массивам. Начальное значение переменной записывается в виде константы соответствующего типа. Начальное значение массива записывается в виде списка констант соответствующего типа. Список констант должен иметь вид:

$$c_1, c_2, \ldots, c_p$$

Здесь c_j ($j = 1, 2, \ldots, p$) — либо константа, либо константа с повторителем.

Константа с повторителем имеет вид:

*r***c*

Здесь c — константа; r — повторитель (целое число без знака), означающий количество повторений константы c. Так, например, запись 3*1.52 равносильна записи 1.52,1.52,1.52. Каждая из констант списка представляет собой начальное значение соответствующего элемента массива. Соответствие между константами и элементами массива устанавливается по порядку их следования в списке констант и внутри массива.

Количество элементов в списке x_i, относящемся к массиву a_i, может быть меньше количества элементов массива a_i. В этом случае перечисленные в списке x_i константы рассматриваются как начальные значения первых элементов массива a_i.

Приведем несколько примеров:

1. Оператор

```
INTEGER*2A1*4(5)/3,4,3*0/,A2/341/,A3,
      A4*4,A5(5,6)
```

говорит о том, что:

а) величины A1, A2, A3, A4 и A5 относятся к целому типу;

б) величина A1 является одномерным массивом стандартной длины с количеством элементов, равным 5. Эти элементы имеют следующие начальные значения:

A1(1) = 3, A1(2) = 4, A1(3) = 0, A1(4) = 0, A1(5) = 0;

в) величина A2 является переменной нестандартной длины с начальным значением, равным 341;

г) величина A3 может быть переменной, массивом или функцией. Длина этой величины нестандартная;

д) величина A4 может быть переменной, массивом или функцией. Длина этой величины стандартная;

е) величина A5 является двумерным массивом с элементами нестандартной длины. Количество измерений — два, максимальные значения первого и второго индексов этого массива соответственно равны 5 и 6.

2. Оператор

```
REAL X1,X2*8,X3(2,3)/1.5,7.01/
```

говорит о том, что:

а) величины X1, X2 и X3 относятся к вещественному типу;

б) величина X1 имеет стандартную длину (по умолчанию) и может быть переменной, массивом или функцией;

в) величина X2 имеет нестандартную длину и может быть переменной, массивом или функцией;

г) величина X3 имеет стандартную длину, является двумерным массивом. Первый и второй элементы этого массива имеют следующие начальные значения:

X(1,1)=1.5, X(2,1)=7.01

Значения других элементов массива не заданы.

3. Оператор

```
LOGICAL Y1,Y2,Y3(3)/.TRUE.,.TRUE.,.FALSE./
```

говорит о том, что:

а) величины Y1, Y2 и Y3 относятся к логическому типу и имеют стандартную длину (по умолчанию);

б) величины Y1 и Y2 могут быть переменными, массивами или функциями;

в) величина Y3 является одномерным массивом, элементы которого имеют следующие начальные значения:

Y3(1)=.TRUE., Y3(2)=.TRUE., Y3(3)=.FALSE.

Для задания начальных значений величин всех типов могут использоваться символьные и шестнадцатеричные константы. Так, например, можно написать такой оператор:

```
INTEGER K/4HX=15/,D*2/'**'/,B/Z00000005/,
        C*2/Z0005/
```

Символьные и шестнадцатеричные константы, используемые для задания начальных значений величин, могут содержать символов меньше, чем требуется для представления начальных значений этих величин. В процессе трансляции программы символьные константы будут дополнены до нужного количества символов пробелами справа, а шестнадцатеричные константы — шестнадцатеричными нулями слева.

Величины, входящие в списки формальных параметров подпрограмм и в операторы общих областей, не могут получать начальные значения через явные операторы типа.

Указатель размерности массива в явном операторе типа можно не задавать. В этом случае он должен быть указан в операторе размеров или в операторе общих областей.

Необходимо следить за тем, чтобы один и тот же индентификатор включался в явные операторы типа не более одного раза.

Вопросы и упражнения

1. Какие описания можно выполнить с помощью явного оператора типа?

2. Как задать тип переменной (функции, массива)?

3. Как задать длину значений переменной (функции, элементов массива)?

4. Может ли число членов списка констант, определяющих начальные значения массива, быть меньше количества членов массива?

5. Можно ли начальное значение переменной задать с помощью явного оператора типа, если эта переменная входит в состав списка формальных параметров некоторой подпрограммы фортран-программы?

6. Указать, какие из приведенных ниже явных операторов типа не содержат ошибок:

1) INTEGER *1X,S(2)/34,3/
2) REAL *4A(3,4),Q,*8Y
3) LOGICAL X(2)/.TRUE.,.TRUE./
4) COMPLEX J1/3.1,(—6.4,0.E0),(2.1E—3,16.7)/,*16P1

7. Указать тип, длину и значения переменных (функций элементов массива), заданных в приведенных ниже явных операторах типа:

1) COMPLEX X2(2)/(—6.4,17.9), (3.7,.96E—2)/,
L*16,S

2) LOGICAL I*1/.FALSE./, J(3)/.TRUE.,.TRUE./

8. Составить явные операторы типа для величин, используемых в следующих выражениях:

1) —4.3*S—X**2/(S—1)+Q(S)

если S и Q — соответственно переменная и функция вещественного типа стандартной длины, X — переменная целого типа нестандартной длины;

2) 1.+(0.7E—2*F+D)*C+C1+EXP(X)**(J—A(I))/B(K)

где F, C и C1 — переменные вещественного типа стандартной длины; D — переменная вещественного типа нестандартной длины; A — одномерный массив трех элементов, значениями которых являются соответственно числа: —3,7; 16,27; -10^2 и B — одномерный мас-

сив пяти элементов, значениями которых являются соответственно числа: $89{,}3 \cdot 10^{-3}$; 0.4; $-16.977 \cdot 10^{-4}$; 0,999; 10^{-15}.

3.1.2. Неявный оператор типа. *Неявный оператор типа* имеет вид:

IMPLICIT $t_1 s_1 c_1, t_2 s_2 c_2, \ldots, t_k s_k c_2$,

где t_i ($i = 1, 2, \ldots, k$) — указатели типа; s_i ($i = 1, 2, \ldots, k$) — указатели длины (указатели типа и длины имеют то же представление, что и в явном операторе типа c_i ($i = 1, 2, \ldots, k$) — *список указателей величин*. Запись $t_i s_i c_i$ означает, что величины, заданные списком c_i, имеют тип t_i и указатель длины s_i. Параметр s_i может отсутствовать. В этом случае считается, что длина стандартная.

Список указателей величин определяет собой группу величин путем перечисления букв, с которых начинаются имена этих величин. Этот список имеет вид:

$(a_1, a_2, \ldots, a_n)$

где a_i ($i = 1, 2, \ldots, n$) — либо отдельная буква, либо *указатель последовательности букв*. Указатель последовательности букв имеет вид:

b_1-b_2

где b_1 — первая, b_2 — последняя буквы последовательности букв, расположенных в алфавитном порядке. Так, например, указатель последовательности букв A-D определяет собой буквы A,B,C,D.

Пример неявного оператора типа:

```
IMPLICIT INTEGER*2(X,K,M),REAL(A-C,P),
         LOGICAL (Y)
```

Этот оператор содержит сведения о том, что величины (переменные, массивы и функции), имена которых начинаются с букв X, K, M, относятся к целому типу и имеют нестандартную длину; величины, имена которых начинаются с букв A, B, C, P, относятся к вещественному типу и имеют стандартную длину; величины, имена которых начинаются с буквы Y, относятся к логическому типу и имеют стандартную длину.

Если тип и длина некоторой величины указаны двумя способами: с помощью явного оператора типа и с помощью неявного оператора типа, то предпочтение отдается явному оператору типа. Это означает, что величина считается того типа и той длины, как это было указано в явном операторе типа. Если же типы и длины некоторых величин, используемых в программе, не были указаны

ни в явном операторе типа, ни в неявном операторе типа, то они определяются по предварительному соглашению (по умолчанию), которое заключается в следующем: величины, имена которых начинаются с букв I, J, K, L, M, N, относятся к целому типу и имеют стандартную длину; величины, имена которых начинаются с других букв, относятся к вещественному типу и имеют стандартную длину.

Вопросы и упражнения

1. С какой целью используется неявный оператор типа?

2. В чем состоят отличия неявного оператора типа от явного оператора типа?

3. Какие средства используются в неявном операторе типа для сокращения перечисления букв в списке указателей величин L, M, N, O, P, Q, R, S?

4. Определить тип и длину переменной SUM, заданную операторами

```
IMPLICIT  REAL(X,S), INTEGER*2(A,E,F),
          COMPLEX SUM
```

5. Указать, какие из приведенных ниже операторов не содержат ошибок:

1)
```
IMPLICIT (S,Q,R),REAL(A,B),
         LOGICAL*1(X)
```
2)
```
IMPLICIT REAL*8(J,I),REAL*2(B-E,F)
         COMPLEX(A)
```
3) `IMPLICIT REAL (A-B,D),INTEGER(J,I), REAL*8(A)`

6. Определить тип и длину переменных, функций и массивов X, Z,S1,S2,S12,BS,D1,D2,J,K,K1,K2,L,C,C1,C2,A(X), используемых в фортран-программе, если в этой программе заданы приведенные ниже операторы:

```
IMPLICIT  LOGICAL(Z,T), REAL(A-D,S),
          INTEGER (F,K)
REAL*8K1,L(2,3)
COMPLEX C*16,S23
```

и других операторов описания типов нет.

3.2. Оператор размеров

Как указывалось в п. 3.1.1, размеры массивов можно указать в явном операторе типа. Однако размеры массивов можно задать и в операторах размеров. Особенно это удобно, когда операторы они-

сания типа отсутствуют, т. е. когда тип и длина всех переменных, функций и массивов определяются по умолчанию.

Оператор размеров имеет вид:

DIMENSION $a_1(k_1), a_2(k_2), \ldots, a_n(k_n)$

где a_i ($i = 1, 2, \ldots, n$) — имя массива, k_i — указатель размерности массива a_i, который имеет ту же структуру, что и в явном операторе типа.

Например,

DIMENSION M1(23),ALFA(15,15),B(3,3,8)

Из этого оператора следует, что M1 — одномерный массив с максимальным значением индекса, равным 23; ALFA — двумерный массив с максимальным значением каждого индекса, равным 15; B — трехмерный массив, в котором максимальные значения первого и второго индексов равны 3 и максимальное значение третьего индекса равно 8.

Если оператор входит в состав подпрограммы и массив a_i, указанный в этом операторе, является формальным параметром, то в качестве максимальных значений индексов этого массива могут употребляться простые переменные целого типа, имеющие стандартную длину. Если массив входит в общую область, то его размеры можно описывать в операторе общих областей.

Вопросы и упражнения

1. Можно ли с помощью оператора размеров описать размерность массива, если его тип и длина ни в каком из операторов фортран-программы не описаны?

2. В каком случае в указателе размерности массива может быть задана простая переменная целого типа в качестве максимального значения индекса?

3. Пусть в фортран-программе задан оператор

DIMENSION A (2,1,3),FOR(3),I(5,2),C(6),
ALPHA (4,3,4),B(4)

Других операторов размера, а также операторов описания типа в этой программе нет.

Определить: а) тип и длину величин, используемых в фортран-программе и имеющих имена J,A30, B, P, A, FOR,F, C, P102, Q, I, G, FQ, ALPHA, X, Y, AL; б) количество элементов в массивах, используемых в программе.

4. Составить операторы описания типа и операторы размера для приведенных ниже переменных, функций и массивов, используемых в программе:

A1,B,CS2,CS — переменных вещественного типа стандартной длины (начальное значение CS равно числу $3{,}4 \cdot 10^{-6}$);

R1,R2,Z,S1,S2 — переменных вещественного типа нестандартной длины;

J1,J2,M,V — переменных целого типа нестандартной длины (начальные значения переменных J1 и M соответственно равны числам 6 и 17);

SIN — функция вещественного типа стандартной длины;

RI — одномерного массива, элементами которого являются соответственно числа: $0{,}6 \cdot 10^{-3}$; $0{,}6 \cdot 10^{-3}$; $0{,}6 \cdot 10^{-3}$; 24,1; 6,7; 6,7; 6,7; 6,7; 15,0;

P2 — двумерного массива целого типа стандатной длины, первый индекс которого принимает значения 1,2,3,4, а второй — 1,2,3.

3.3. Оператор эквивалентности

Оператор эквивалентности имеет вид:

$$\text{EQUIVALENCE } (a_1^1, a_2^1, \ldots, a_{n_1}^1), (a_1^2, a_2^2, \ldots, a_{n_2}^2), \ldots$$
$$\ldots, (a_1^k, a_2^k, \ldots, a_{n_k}^k),$$

где a_i^j ($j = 1,2,\ldots, k$; $i = 1,2,\ldots, n_j$) — либо простая переменная, либо элемент массива с численными индексами, либо элемент массива, который задается именем массива и порядковым номером элемента в массиве, заключенным в круглые скобки и помещенным непосредственно за именем массива.

Запись $(a_1^i, a_2^i, \ldots, a_{n_i}^i)$ называется *группой эквивалентности* и означает, что перечисленные в скобках переменные должны быть совмещены, т. е. расположены в памяти с одного и того же места (заметим, что величины, не указанные в операторах эквивалентности, размещаются в индивидуальных областях памяти). Эти переменные могут относиться к различным типам и иметь различную длину.

Так, например, оператор

```
EQUIVALENCE (A,B(3),D(8)), (E,F(5))
```

указывает на то, что переменная A, третий элемент массива B и восьмой элемент массива D должны быть совмещены по памяти.

Кроме того, должны быть совмещены по памяти переменная E и пятый элемент массива F. Так как элементы массивов хранятся в последовательных ячейках памяти, то совмещение по памяти одних элементов массивов может привести к совмещению других элементов этих массивов.

Приведем такой пример: пусть имеется два одномерных массива A и B, которые относятся к целому типу и имеют одинаковую длину. Пусть далее задан оператор эквивалентности

EQUIVALENCE (A(4), B(2)).

В соответствии с указаниями этого оператора массивы A и B будут размещены в памяти таким образом, чтобы элемент A (4) был совмещен с элементом B (2). Это приведет к тому, что элемент A (3) будет совмещен с элементом B (1), элемент A (5) с элементом B (3) и т. д.

Большинство трансляторов с языка фортран требует, чтобы переменные, указанные в группе эквивалентности, размещались в памяти ЭВМ с нужных границ. Поясним это обстоятельство.

Основная память машин ЕС ЭВМ состоит из восьмиразрядных ячеек, называемых *байтами* памяти. Байты имеют сквозную нумерацию. Под каждую из величин в зависимости от ее типа и длины отводится определенное количество байтов. Так, под величину логического типа нестандартной длины отводится один байт (однобайтовая ячейка), под величину целого типа нестандартной длины — два байта (двухбайтовая ячейка, или *полуслово*), под величину целого типа стандартной длины, вещественного типа стандартной длины и логического типа стандартной длины — четыре байта (четырехбайтовая ячейка, или *слово*), под величину вещественного типа нестандартной длины — восемь байтов (восьмибайтовая ячейка, или *двойное слово*), под величину комплексного типа стандартной длины — восемь байтов (две четырехбайтовые ячейки, или двойное слово), под величину комплексного типа нестандартной длины — шестнадцать байтов (две восьмибайтовые ячейки или два двойных слова).

Система команд ЭВМ требует, чтобы адрес первого байта полуслова был кратным двум, адрес первого байта слова был кратным четырем, адрес первого байта двойного слова был кратным восьми. Адрес однобайтовой ячейки может быть любым.

Размещением величин в ячейках памяти с нужными адресами их первых байтов (с нужных границ) занимается транслятор и разработчику программы об этом заботиться, вообще говоря, не нужно. Однако при написании операторов эквивалентности можно создать такую ситуацию, когда транслятор не сможет разместить не-

которые величины с нужных границ. Чтобы это избежать, надо учитывать правила распределения памяти под величины, входящие в группу эквивалентности. Рассмотрим эти правила:

1. Если группа эквивалентности содержит элемент массива, то считается, что весь этот массив входит в данную группу эквивалентности.

2. Под группу эквивалентности выделяется поле памяти такое, чтобы в нем могли разместиться значения всех величин, входящих в данную группу эквивалентности (с учетом указанных совмещений).

3. Поле, выделенное под группу эквивалентности, размещается с границы двойного слова.

Чтобы определить, с какой границы размещается данная величина, достаточно знать ее относительный адрес, который отсчитывается от первого байта поля, отведенного под группу эквивалентности. Первый байт поля имеет нулевой относительный адрес и может служить границей двойного слова, слова, полуслова и однобайтовой ячейки. Если относительный адрес величины кратен восьми, то величина размещается с границы двойного слова; если не кратен восьми, но кратен четырем, то с границы слова и т. д. Заметим, что граница двойного слова является границей слова, полуслова и однобайтовой ячейки, граница слова — границей полуслова и однобайтовой ячейки и т. д.

Приведем примеры:

1) Пусть A — массив вещественного типа стандартной длины, содержащий 10 элементов; B — переменная вещественного типа стандартной длины. Тогда в соответствии с указаниями оператора

EQUIVALENCE (B,A(4))

под содержащуюся в нем группу эквивалентности будет отведено 40 байтов. Относительные адреса величин, входящих в эту группу, приведены в табл. 3.1. Все величины этой группы размещаются с нужных границ.

2) Пусть A — массив целого типа нестандартной длины, содержащий 20 элементов; B — массив комплексного типа стандарт-

Таблица 3.1. Относительные адреса величин, содержащихся в группе эквивалентности (B, A(4))

Величина	A(1)	A(2)	. . .	A(10)	B
Относительный адрес	0	4	. . .	36	12

ной длины, содержащий 10 элементов; C — переменная вещественного типа нестандартной длины. Тогда в соответствии с указаниями оператора

EQUIVALENCE (A(3),B(1),C)

под содержащуюся в нем группу эквивалентности будет отведено 84 байта. Относительные адреса величин, входящих в группу эквивалентности, приведены в табл. 3.2.

Таблица 3.2. Относительные адреса величин, содержащихся в группе эквивалентности (A(3), B(1), C)

Величина	Относительный адрес	Величина	Относительный адрес	Величина	Относительный адрес
A(1)	0	B(1)	4	C	4
A(2)	2	B(2)	12		
A(3)	4	B(3)	20		
⋮	⋮	⋮	⋮		
A(20)	18	B(10)	76		

Заметим, что все величины, за исключением величины C, размещаются с нужных границ. Величина C занимает в памяти двойное слово, а размещается с границы слова.

Оператор эквивалентности не должен содержать противоречивых требований как внутри себя, так и по отношению к ранее установленным эквивалентностям.

Оператор эквивалентности относится к числу невыполняемых операторов.

Вопросы и упражнения

1. Указать, какие из перечисленных ниже операторов эквивалентности не содержат ошибок:

1) EQUIVALENCE (A,B), (X,Y)
2) EQUIVALENCE (A,B,C(2))
3) EQUIVALENCE (X(I),Y), (S,Q)
4) EQUIVALENCE (X(2),B,X(4))
5) EQUIVALENCE (A(3),B),(A(5),C)
6) EQUIVALENCE (A,X,X+A)

2. Пусть задан оператор эквивалентности EQUIVALENCE (A,B). Определить значение величины B, если известно, что A = = —3.7 и A, B имеют одинаковые тип и длину.

3. Пусть элементы двумерного массива L имеют следующие значения:

L(1,1)=—10^{-3}; L(2,1)=4,6; L(3,1)=0,0;
L(1,2)=—0,5·10^{3}; L(2,2)=8,9; L(3,2)=0,1.

Определить значение переменной F, если задан оператор

EQUIVALENCE (F,L(5)).

4. Пусть A1 — переменная вещественного типа стандартной длины, X — одномерный массив целого типа стандартной длины, состоящий из 6 элементов. Определить количество единиц памяти (байтов), которое потребуется для размещения группы эквивалентности, заданной оператором

EQUIVALENCE (A1,X(3)).

5. Определить относительные адреса величин, входящих в группу эквивалентности, заданной оператором эквивалентности

EQUIVALENCE (X,Z(4),S(3),A(2)),

если известно, что X — переменная вещественного типа стандартной длины; Z — массив комплексного типа нестандартной длины, содержащий 4 элемента; S — массив вещественного типа нестандартной длины, содержащий 22 элемента; A — массив вещественного типа стандартной длины, содержащий 17 элементов.

3.4. Оператор общих областей

В языке фортран предусмотрена возможность размещать величины, используемые в разных модулях фортран-программы, в одни и те же области оперативной памяти, которые принято называть *общими областями*.

Общей области можно поставить в соответствие некоторое имя (идентификатор). Такую область принято называть *именованной*. Однако общая область может и не иметь имени. Такая область называется *неименованной*.

Для создания общих областей путем указания имен этих областей и списков величин, которые включаются в эти области, используется оператор общих областей, который относится к числу невыполняемых операторов.

Оператор общих областей имеет вид:

COMMON /r_1/c_1, /r_2/c_2, . . ., /r_n/c_n

где r_i — либо имя общей области (для именованной области), либо пусто (для неименованной области); c_i ($i = 1,2,\ldots, n$) — список величин, помещаемых в общую область с именем r_i, либо в неименованную область.

Если список c_1 относится к неименованной области, то элемент r_1 можно опустить.

Список величин, помещаемых в общую область, имеет вид:

$$b_1, b_2, \ldots, b_p$$

Здесь b_j ($j = 1,2,\ldots, p$) — имя переменной, имя массива или имя массива с указателем размерности. Указатель размерности массива пишется только в том случае, если он не был указан ни в операторе размеров, ни в явном операторе типа.

Порядок размещения величин в общих областях определяется порядком их перечисления в операторах общих областей.

В одном модуле фортран-программы может быть несколько операторов общих областей. Совокупность общих областей и перечень величин, входящих в каждую такую область, определяется всем набором операторов общих областей, а порядок следования величин в каждой из общих областей определяется порядком следования операторов общих областей в программном модуле.

Приведем несколько примеров.

1. Если задан один оператор

```
COMMON X,Y,Z (10)
```

то это означает, что будет создана одна неименованная общая область и в ней разместятся величины X, Y, Z в заданном порядке.

2. Если заданы два оператора

```
COMMON X,Y,Z (10)
COMMON A,B,C,D
```

то это означает, что будет создана одна неименованная общая область и величины в ней будут размещены в следующем порядке:

```
X,Y,Z,A,B,C,D
```

3. Если задан оператор

```
COMMON X1,Y1,/R1/Z(5,15),W,//X2,Y2
```

то будут созданы две общие области: неименованная (в нее войдут величины X1, Y1, X2, Y2) и с именем R1 (в нее войдут величины Z, W).

4. Если заданы операторы

```
COMMON A1,B1,/R1/C1,D1,//X1,Y1
COMMON /R1/A2,B2,//C2,D2,/R2/X2,Y2
```

то будут созданы три общие области: неименованная (в нее войдут величины A1,B1,X1,Y1,C2,D2); с именем R1 (в нее войдут величины C1,D1,A2,B2) и с именем R2 (в нее войдут величины X2,Y2).

Одноименные общие области, относящиеся к разным программным модулям, будут совмещены друг с другом, т. е. будут размещены с одного и того же места памяти. Это позволяет экономить расход внутренней памяти и осуществлять обмен данными между различными программными единицами одной и той же программы (неименованные общие области считаются одноименными).

Требования к одноименным общим областям различных программных модулей одной и той же программы:

1) Размеры неименованных общих областей во всех программных модулях должны быть одинаковыми.

Однако это не означает, что все программные модули одной и той же программы должны иметь неименованные общие области. Некоторые из них могут ее не иметь.

2) Если общие области используются для обмена данными между двумя программными единицами, то типы и длины данных, входящих в общую область программной единицы, должны совпадать с типами и длинами соответствующих данных, входящих в общую область другой программной единицы. Соответствие между данными устанавливается по порядку их следования в общих областях.

3) Величины, входящие в неименованные общие области, не могут иметь начальных значений. Начальные значения величин, входящих в именованные общие области, могут задаваться только в подпрограммах начальных данных.

Операторы общих областей и операторы эквивалентности должны быть согласованы между собой, т. е. они не должны содержать противоречивых требований. Так, например, нельзя совмещать две переменные, которые входят в одну или разные общие области. В то же время одна или несколько переменных, не входящих ни в одну из общих областей, могут быть совмещены с переменной, входящей в одну из общих областей. При согласовании операторов эквивалентности и общих областей полезно помнить о том, что общие области создаются по времени раньше, чем осуществляется совмещение величин.

Если элемент массива, не входящего в общую область, совмещается с величиной, входящей в общую область, то при этом общая

область может расшириться. Расширения допускаются только в сторону смещения конца общей области.

Большинство трансляторов с языка фортран требует, чтобы величины в общих областях памяти размещались не в произвольном порядке, а с нужных границ. Правила определения границ были рассмотрены при описании оператора эквивалентности.

Нужные границы для величин, входящих в общие области, можно обеспечить путем их размещения в этих областях по убывающим длинам: вначале размещаются величины, значения которых занимают по 8 байтов, за ними размещаются величины, значения которых занимают по 4 байта и т. д. Кроме того, для выравнивания границ можно рекомендовать включение в общие области фиктивных переменных.

Вопросы и упражнения

1. Для каких целей используется оператор общих областей?
2. В каком случае для массива, размещаемого в общей области, задается указатель размерности в операторе общих областей?
3. В каком порядке размещаются величины в общей области?
4. Могут ли в разных программных модулях задаваться операторы общих областей с одинаковыми именами общих областей?
5. Перечислить требования к одноименным общим областям различных программных модулей одной и той же фортран-программы.
6. В каком случае допускается расширение общей области?
7. Указать имена общих областей и порядок размещения величин в общих областях, заданных следующими операторами:

1) COMMON X,Y,Z,/W/A(50),B(70,40),//C,/W/C1,C2(2),//X1,X2,
/S/Q,R,T

2) COMMON/R1/A,B,C,//X,Z,/R2/X1,X2(10),//T,T1,T2(5),
/R1/S,A1,A2
COMMON F,E,J,/R2/K,L,M(10)

8. Указать, какие из перечисленных ниже операторов общих областей и эквивалентности допустимы, если считать, что все перечисленные в них величины относятся к вещественному типу стандартной длины и все массивы содержат указатели размеров.

1) COMMON A,B,C,D(50)
EQUIVALENCE (X,E(5),Y)

2) COMMON /R1/A1,A2,A3,A4,/R2/B1,B2,B3,B4,B5
EQUIVALENCE (X1,A1,C1), (Y1,B2,D)

3) COMMON /R1/A1,A(2),A3,/R2/B1,B2,B3
EQUIVALENCE (A1,B1),(C,D)

4) COMMON /R1/A1,A2,X(50),/R2/B1,B2,Y(50)
EQUIVALENCE (Y(3),X(1)), (Z(1), Y(35))

9. Пусть заданы массивы A(50), B(7,9), C(10) и переменные X,Y,Z,P,Q. Написать операторы, размещающие:

1) величины A,C,Z в неименованную общую область, а величины B,X,Y,P и Q в общую область с именем K;

2) массивы A,B и C в неименованную общую область так, чтобы переменным A (1), B (1) и C (1) была выделена одна и та же область памяти;

3) величины A,B,X, Y,Z в общую область с именем K так, чтобы переменной X была бы выделена та же область памяти, что и переменной A (5), переменной Z та же область памяти, что и переменной Y.

3.5. Оператор-функция

Этот оператор представляет собой один из способов задания процедур вычисления значений функций. Он имеет вид:

$$f(x)=a$$

где f — имя (идентификатор) функции; x — *список формальных параметров оператора-функции*; a — арифметическое или логическое выражение.

Список формальных параметров имеет вид:

$$x_1,x_2,\ldots,x_n$$

Здесь x_i $(i = 1,2,\ldots, n)$ — формальные параметры.

Требования к формальным параметрам:

1. В списке формальных параметров должен присутствовать хотя бы один формальный параметр.

2. В качестве формальных параметров могут использоваться только простые переменные.

3. Все формальные параметры должны быть попарно различными.

4. Формальные параметры одних операторов-функций могут использоваться в качестве формальных параметров других операторов-функций.

5. Для указания типов и длин формальных параметров используются обычные средства (неявные и явные операторы типа, предварительное соглашение).

6. Вне операторов-функций формальные параметры могут использоваться как обычные переменные.

В состав выражения *a* оператора-функции могут входить формальные параметры, простые переменные, не являющиеся формальными параметрами данного оператора-функции, а также указатели функций. Однако оператор-функция не должен содержать прямых или косвенных (через другие операторы-функции) обращений к самому себе.

Так, например, недопустимы следующие операторы:

```
1) F (X) = X**2 + Y**2 + F (Z)
2) F1 (X) = A*X**3 + B*X**2 + F2 (X)
   F2 (X) = SIN (X) + COS (X) + F1 (B)
```

Оператор-функция, приведенный в первом примере, содержит обращение непосредственно к самому себе, а операторы-функции, приведенные во втором примере, содержат обращение к самим себе не непосредственно, а через другие операторы-функции.

Если выражение *a*, входящее в состав оператора-функции, является арифметическим, то функция *f* должна относиться к одному из арифметических типов; если же выражение *a* является логическим, то функция *f* должна относиться к логическому типу. Для задания типа и длины функции *f* используются обычные средства (неявные и явные операторы типа, предварительное соглашение).

Примеры операторов-функций:

```
F (X, Y) = (SQRT (X) — X1)**2 + (Y — Y1)**2
DET (L) = L.GE.A.AND.L.LE.B
SC2 (X) = (SIN (X) + COS (X))/2
```

Выражение *a*, стоящее в правой части оператора-функции, определяет операции, которые нужно выполнить при обращении к вычислению функции.

Оператор-функция относится к числу выполняемых операторов. Однако обращение к оператору-функции можно осуществить только с помощью указателя функции. При этом должны выполняться следующие условия:

1. Количество фактических параметров указателя функции должно совпадать с количеством формальных параметров оператора-функции.

2. Типы и длины фактических параметров должны совпадать с типами и длинами соответствующих формальных параметров. Соответствие устанавливается по порядку их следования в списках параметров (первому формальному параметру соответствует первый фактический параметр, второму формальному параметру — второй фактический параметр и т. д.).

3. В качестве фактических параметров могут использоваться константы, переменные, а также выражения.

4. Указатель функции, посредством которого осуществляется обращение к вычислению функции, должен входить в состав той же программной единицы, что и соответствующий ему оператор-функция.

Вычисление значения функции осуществляется следующим образом:

1) Формальные параметры оператора-функции, входящие в состав выражения *a*, заменяются на соответствующие фактические параметры указателя функции *f*.

2) Вычисляется значение выражения *a*, которое затем приводится к типу и длине функции *f* и присваивается указателю функции, с помощью которого осуществлялось обращение к вычислению этой функции.

Приведем пример. Пусть имеется оператор-функция

```
F(X,Y)=3.0*X+Y**2+X+Y+Z
```

и арифметический оператор присваивания

```
Z=F(D,E)+K
```

Действие этих операторов будет эквивалентно действию следующих двух операторов присваивания:

```
F1=3.0*D+E**2+D+E+Z
Z=F1+K
```

Здесь F1 — переменная того же типа и той же длины, что и функция F.

Функции, процедуры вычисления которых задаются при помощи операторов-функций, принято называть *операторными функциями*. В программных модулях они размещаются перед первым выполняемым оператором.

Вопросы и упражнения

1. Перечислить требования, предъявляемые к формальным параметрам операторов-функций.

2. Могут ли в состав выражения операторов-функций входить переменные, отличные от формальных параметров?

3. Какими средствами языка фортран можно задать свойства (тип и длину) функции?

4. В какой части фортран-программы должны размещаться операторы-функции?

5. В каком случае может начать выполняться оператор-функция?

6. Какие действия выполняются при вычислении значения функции?

7. Указать, какие из приведенных ниже операторов-функций содержат ошибки:

1) S(X, Y) = (X + 3 * Y)/2 + A — 3.7
2) L(X, Y, X) = (X — Y * * 2) * X — 3 * Y + C
3) A(S, F(J)) = —4.7 + S — S * * 2 * F(J)/2
4) T(X, Y) = X — T(X, Y) + 2 * X * T (X, Y)

8. Определить значение величины Z, которое она получит в результате выполнения следующих операторов:

```
F(X, Y) = (X + Y)/2
A = —3.4
B = 0.1
Z = F(A, B) + A/B
```

3.6. Оператор конца

Этот оператор имеет вид:

```
END
```

и служит для обозначения конца основной программы и подпрограмм. *Оператор конца* должен быть последним оператором в программном модуле. Следует особо отметить, что он является невыполняемым оператором и не может использоваться для указаний об окончании выполнения программы (для этой цели служат операторы возврата и останова).

3.7. Основная программа

Основная программа имеет вид

s_1
s_2
.
.
.
s_n

где s_n — оператор конца; s_i ($i = 1, 2, \ldots, n-1$) — последовательность выполняемых и невыполняемых операторов, не содержащая заголовков подпрограмм и операторов конца.

При написании последовательности операторов s_i необходимо соблюдать следующие требования:

1. Последовательность s_i должна содержать не более одного неявного оператора типа, причем таким оператором может быть только оператор s_1.

2. Операторы, описывающие данную величину (операторы описания типа, операторы размеров, эквивалентности, общих областей), должны размещаться раньше операторов, использующих эту величину.

3. Операторы-функции должны размещаться раньше первого выполняемого оператора программы.

4. Операторы-функции должны размещаться друг относительно друга в такой последовательности, чтобы не было обращений к операторам-функциям, расположенным позже данного оператора-функции (т. е. позже оператора-функции, из которого осуществляется обращение). Так, например, последовательность операторов

```
F1(X,Y)=X**2+Y**2
F2(X,Y)=F1(X,Y)+X**Y
```

допустима, а последовательность

```
F2(X,Y)=F1(X,Y)+X**Y
F1(X,Y)=X**2+Y**2
```

не допустима.

Выполнение основной программы начинается с первого выполняемого оператора и должно завершаться либо оператором останова, либо оператором возврата.

Рассмотрим пример на составление основной программы. Пусть задана последовательность вещественных чисел $N_1, N_2, \ldots, N_k$ ($k \leqslant 100$). Требуется составить программу, которая определяла бы количество положительных и количество отрицательных чисел, содержащихся в этой последовательности. Число нуль не относить ни к положительным, ни к отрицательным. Числа $N_1, N_2, \ldots, N_k$ должны быть представлены с точностью до 12 десятичных знаков. Так как операторы ввода и вывода нами еще не рассматривались, то ввод исходных данных и вывод результатов в программе не предусматривать. Ниже приведен один из возможных вариантов такой программы.

```
      REAL *8N(100)
      INTEGER  K,M1/0/,M2/0/
      DO 3 I=1,K
      IF (N(I))1,3,2
    1 M2=M2+1
      GO TO 3
```

```
    2 M1=M1+1
    3 CONTINUE
      STOP
      END
```

Вопросы и упражнения

1. Может ли основная программа заканчиваться оператором, отличным от оператора конца?

2. Какой порядок операторов принят в основной программе?

3. Могут ли в состав основной программы входить подпрограммы?

4. Определить значения величин Z1 и Z2, которые они получат в результате выполнения следующей программы:

```
SUM(X,Y)=(X+Y)/Y
P(X,B)=X*B/2
X=0.0
B=X+3.4
Y=-2.3
Z1=SUM(X,Y)
Z2=P(X,Y)
STOP
END
```

5. Составить программу вычисления значений величины

$$x_i = f(y_i) + 0.6 \cdot 10^{-3},$$

где

$$y_i = z_i + 2,$$

$$f(y_i) = \begin{cases} y_i^2 - 0.3, & \text{если } y_i < 0 \\ 0, & \text{если } 0 \leqslant y_i \leqslant 1 \\ y_i^2 + y_i, & \text{если } y_i > 0 \end{cases}$$

для всех $z_i = z_{i-1} + h$, $z_0 = -1$, $h = 0.5$, $(i = 1,2,\ldots 10)$.

6. Составить программу вычисления логической переменной D, принимающей значение истина, если точка A с координатами X и Y принадлежит внутренней области круга $(X-X_0)^2+(Y-Y_0)^2$, и значение ложь — в противном случае. В программе предусмотреть оператор-функцию для вычисления функции $F(X,Y)=(X-X_0)^2+(Y-Y_0)^2$.

7. Может ли оператор-функция, содержащий обращения к другому оператору-функции, стоять в программе раньше, чем этот второй оператор-функция?

Глава 4

ПОДПРОГРАММЫ

4.1. Подпрограмма-функция

Подпрограмма-функция служит для вычисления значений некоторой функции, связанной с данной подпрограммой. Функция, вычисление значений которой осуществляется по подпрограмме, называется *внешней функцией*. Подпрограмма-функция имеет вид:

s_0
s_1
.
.
.
s_n

где s_0 — заголовок функции; s_i ($i = 1, 2, \ldots, n-1$) — последовательность выполняемых и невыполняемых операторов, не содержащая заголовков функций, заголовков процедур и операторов конца; s_n — оператор конца.

Последовательность операторов s_i должна удовлетворять тем же требованиям, что и последовательность операторов основной программы.

4.1.1. Заголовок функции. *Заголовок функции* имеет вид:

t FUNCTION $f s(x)$

где t — указатель типа функции; f — имя функции (идентификатор); s — указатель длины функции; x — список формальных параметров.

Указатель типа и указатель длины рассмотрены при описании явных и неявных операторов типа (см. п. 3.1). Указатели типа и длины могут отсутствовать. В этом случае тип и длина функции определяются внутри подпрограммы обычными средствами (неявными и явными операторами типа, предварительным соглашением).

Список формальных параметров имеет вид:

$$x_1, x_2, \ldots, x_n$$

где x_i ($i = 1, 2, \ldots, n$) — либо формальный параметр, либо формальный параметр, заключенный в косые черточки. Список формальных параметров должен содержать хотя бы один формальный параметр.

Формальные параметры — это идентификаторы, которые внутри подпрограммы используются как имена простых переменных, массивов, функций и процедур. В косые черточки могут заключаться только те формальные параметры, которые внутри подпрограммы используются как имена простых переменных.

Примеры заголовков функций:

```
INTEGER FUNCTION SUM*2(S,N,M)
FUNCTION F1(X,/Y/)
```

Для задания типа, длины и размерности формальных параметров используются обычные средства (предварительное соглашение, явные и неявные операторы типа, операторы размеров).

Формальные параметры подпрограмм-функций не могут иметь начальных значений и не могут включаться ни в операторы общих областей, ни в операторы эквивалентности.

Вопросы и упражнения

1. Для каких целей используются подпрограммы-функции?
2. Чем отличается структурно подпрограмма-функция от основной программы?
3. Какие составные элементы обязательны для заголовка функции?
4. Какими средствами в подпрограмме-функции можно описать величины (их тип и длину), входящие в состав формальных параметров?
5. Можно ли в подпрограмме-функции формальным параметрам задать начальные значения?
6. Могут ли в подпрограмме-функции формальные параметры входить в состав операторов общих областей и операторов эквивалентности?
7. Указать имя функции, ее тип и длину, а также список формальных параметров в приведенной ниже подпрограмме-функции

```
FUNCTION Q*8(A,N)
DIMENSION A(N)
Q=0
```

```
      DO 1 J=1,N
    1 Q=Q+A(J)+(1+2*A(J))/3
      RETURN
      END
```

Определить тип, длину формальных параметров и размерность массивов, входящих в состав формальных параметров.

4.1.2. Выполнение подпрограммы-функции. Подпрограмма-функция выполняется только при обращении к ней из другого программного модуля с помощью указателя функции. *Фактические параметры* указателя функции должны удовлетворять следующим требованиям:

1. Количество фактических параметров указателя функции должно совпадать с количеством формальных параметров подпрограммы.

2. Типы и длины фактических параметров должны совпадать с типом и длиной соответствующих формальных параметров. Соответствие между параметрами устанавливается по порядку их следования в списках параметров (первому формальному параметру соответствует первый фактический, второму формальному параметру соответствует второй фактический и т. д.).

3. Если формальный параметр — имя функции, то соответствующий фактический параметр должен быть имéнем внешней функции.

4. Если формальный параметр — имя процедуры, то соответствующий фактический параметр должен быть именем процедуры.

5. Если формальный параметр — имя массива, то в качестве соответствующего фактического параметра может указываться либо имя массива, либо переменная с индексом (элемент массива).

6. Если формальный параметр — имя переменной, которая в подпрограмме не получает новых значений, то в качестве соответствующего фактического параметра могут использоваться константы, переменные и выражения. Если же формальный параметр получает новые значения, то соответствующим фактическим параметром может быть только переменная.

7. Если формальным параметром подпрограммы-функции является имя переменной или имя массива, то в качестве соответствующего фактического параметра указателя функции может использоваться символьная константа. Эта константа помещается в оперативную память с границы двойного слова и будет занимать столько байтов, какова длина этой константы. Необходимо следить за тем, чтобы длина этой константы соответствовала длине формаль-

ного параметра. Имя функции должно получать значение внутри подпрограммы функции. В момент выхода из подпрограммы значение, присвоенное имени функции, берется в качестве значения функции. Присваивание значения имени функции можно обеспечить путем включения его в левую часть оператора присваивания, в список вводимых величин оператора ввода, а также в список фактических параметров оператора процедуры или указателя функции. При обращении к подпрограмме-функции фактические параметры указателя функции могут вызываться в подпрограмму как по имени, так и по значению.

Вызов фактических параметров по имени заключается в следующем: перед началом выполнения подпрограммы формальные параметры, входящие в состав операторов подпрограммы, всюду в этой подпрограмме заменяются на соответствующие фактические параметры. В этом случае предусмотренные над формальными параметрами операции в действительности будут выполняться над соответствующими фактическими параметрами.

Вызов фактических параметров по значению заключается в следующем: перед началом выполнения подпрограммы значения фактических параметров присваиваются соответствующим формальным параметрам. После выполнения подпрограммы значения формальных параметров присваиваются соответствующим фактическим параметрам.

Фактический параметр вызывается по имени, если соответствующий ему формальный параметр либо заключен в косые черточки, либо является именем массива, функции или процедуры.

Если формальный параметр является именем переменной и не заключен в косые черточки, то соответствующий ему фактический параметр вызывается по значению.

Если фактический параметр является выражением, то он заменяется на простую переменную, которая является результирующей по отношению к выражению. Тип и длина этой переменной берутся равными типу и длине выражения. При обращении к подпрограмме результирующая переменная играет роль фактического параметра.

Формальные массивы могут иметь так называемые *регулируемые размеры*. Указатели размерностей таких массивов в качестве максимальных значений некоторых индексов содержат простые переменные целого типа стандартной длины, которые называются регулируемыми размерами. Каждый из регулируемых размеров должен входить либо в список формальных параметров подпрограммы-функции, либо в одну из общих областей. В момент обращения к под-

программе-функции регулируемые размеры должны получать конкретные значения, которые должны передаваться из вызывающего программного модуля в первом случае через фактические параметры указателя функции, а во втором — через переменные, входящие в соответствующие общие области. Переменные, относящиеся к регулируемым размерам, внутри подпрограммы-функции не должны менять своих значений.

Выполнение подпрограммы-функции начинается с первого выполняемого оператора и должно завершаться оператором возврата RETURN. Одна подпрограмма-функция может содержать несколько операторов возврата. Подпрограмма-функция не должна содержать прямых или косвенных (через другие подпрограммы) обращений к самой себе.

Примеры подпрограмм-функций:

```
1) REAL FUNCTION F(X)
   REAL X
   F=5*X**2
   RETURN
   END
```

Приведенная подпрограмма-функция позволяет вычислять значения функции $F=5x^2$.

```
2)    INTEGER FUNCTION F5(K)
      DIMENSION K(5)
      N=0
      DO 1 I=1,5
    1 N=N+K(I)
      F5=N
      RETURN
      END
```

Эта подпрограмма-функция позволяет вычислять значения величины $F5=\sum_{i=1}^{5} k_i$

```
3)    FUNCTION DT(N,C1,C2)
      INTEGER C1,C2
      REAL N(C2)
      X=1.0
      DO 1 K=C1,C2
    1 X=X+N(K)
      DT=X
      RETURN
      END
```

Эта подпрограмма-функция позволяет вычислять значения величины

$$DT = 1 + \sum_{k=C_1}^{C_2} n_k$$

```
4) FUNCTION F2(/A2/)
   A2=A2**2
   F2=A2**2+A2
   RETURN
   END
```

Эта подпрограмма-функция позволяет вычислять значение величины $f = a + a^2$.

Если программный модуль использует имя внешней функции (в указателе функции или в списке фактических параметров), то он должен содержать сведения о типе и длине этой функции. Для этой цели используются обычные средства (явные и неявные операторы типа, предварительное соглашение). Имена внешних функций, входящих в списки фактических параметров указателей функций и операторов процедур (см. п. 4.2), должны быть перечислены в операторе внешних подпрограмм, входящем в состав того программного модуля, который содержит эти указатели функций и операторы процедур.

Вопросы и упражнения

1. Какими средствами языка фортран можно обратиться к выполнению подпрограммы-функции?

2. Перечислить требования, которым должны удовлетворять фактические параметры указателя функции.

3. Определить, чем представлен каждый из формальных параметров (переменной, массивом, именем функции или процедуры) подпрограммы-функции, если обращение к ней осуществляется с помощью указателя функции

```
F(X(3),A+2*B,—2.7E—3)
```

Что можно сказать относительно третьего формального параметра?

4. Назвать два способа вызова фактических параметров в подпрограмму. Чем отличаются эти способы?

5. Что называется регулируемыми размерами массивов?

6. Могут ли в состав подпрограммы входить операторы общих областей, и если могут, то какие величины могут представляться в этих операторах?

7. Чем заканчивается выполнение подпрограммы-функции?

8. Определить значение переменной величины N, которое будет вычислено в результате выполнения следующей программы:

```
Основная программа          Подпрограмма-функция
  INTEGER F                 INTEGER FUNCTION F(K)
  DIMENSION M(10)           DIMENSION K(5)
  M(1)=1                    F=0
  DO 2 K=2,10               DO 1 J=1,5
2 M(K)=1+M(K-1)           1 F=F+K(J)
  N=F(M)                    RETURN
  STOP                      END
  END
```

4.1.3. Оператор внешних подпрограмм. Этот оператор имеет вид:

EXTERNAL $a_1, a_2, \ldots, a_n$

где a_i ($i = 1, 2, \ldots, n$) — имена внешних функций и имена процедур, которые входят в списки фактических параметров указателей функции и операторов процедур.

Примеры программ, содержащих подпрограммы-функции.

1) Основная программа

```
  INTEGER F5
  DIMENSION M(10)
  DO 2 K=1,10
2 M(K)=K**2+1
  N=F5(M,10)
  STOP
  END
```

Подпрограмма-функция

```
INTEGER FUNCTION F5(K,N)
DIMENSION K(N)
M=0
DO 1 J=1,N
1 M=M+K(J)
F5=M
RETURN
END
```

2) Основная программа

```
REAL KBN1
X=2
Z=KBN1(X)
RETURN
END
```

Подпрограммы-функции

```
REAL FUNCTION KBN1(A)
REAL KBN2
B=30*A+4.0
KBN1=KBN2(B)
RETURN
END
REAL FUNCTION KBN2(A)
KBN2=5.0*A+6.0
RETURN
END
```

3) Основная программа

```
INTEGER F5
DIMENSION M(10)
COMMON L
DO 2 K=1,10
M(K)=K**2+1
L=10
N=F5(M)
STOP
END
```

Подпрограмма-функция

```
INTEGER FUNCTION F5(K)
DIMENSION K(N)
COMMON N
M=0
DO 1 J=1,N
1 M=M+K(J)
F5=M
RETURN
END
```

4) Основная программа

```
X1=2.0
Y1=F1(X 1)
X2=2.0
Y2=F2(X2)
RETURN
END
```

Подпрограммы-функции

```
FUNCTION F1(A1)
A1=A1**2
F1=A1**2+A1
RETURN
END
FUNCTION F2(A1)
F2=1+A1+A1**2
RETURN
END
```

5) Основная программа

```
REAL A,B,F
A=3.0
B=F(A)—7.5
STOP
END
```

Подпрограмма-функция

```
REAL FUNCTION F(X)
REAL X
F=5*X**2
RETURN
END
```

6) Основная программа

```
REAL N1,N2
EXTERNAL N1,N2
X=2.0
Z1=T(X,N2)
Y=3.0
Z2=T(Y, N1)
STOP
END
```

Подпрограммы-функции

```
FUNCTION T(A,C)
B=A**2
T=C(B)
RETURN
END
REAL FUNCTION N1(X)
N1=3*X
RETURN
END
REAL FUNCTION N2(X)
N2=4*X
RETURN
END
```

4.1.4. Кратный вход в подпрограмму-функцию. Подпрограмма-функция может использоваться для вычисления значений нескольких функций. Одна из них считается *основной*, а остальные — *вспомогательными*. Имя основной функции указывается в заголовке функции, который является основным оператором входа в подпрограмму-функцию. Обращение к подпрограмме за вычислением значения основной функции мы уже рассматривали.

Обращение к подпрограмме-функции за вычислением вспомогательной функции осуществляется с помощью указателя функции $f_i(y_i)$, где f_i — имя i-й вспомогательной функции, y_i — список фактических параметров (аргументов) функции $f_i(y_i)$.

В этом случае выполнение подпрограммы-функции начинается с первого выполняемого оператора, следующего за вспомогательным оператором входа, относящимся к функции f_i.

Вспомогательный оператор входа имеет вид:

ENTRY $f_i(x_i)$

где f_i — имя вспомогательной функции, x_i — список формальных параметров функции f_i.

Тип и длина величины f_i и формальных параметров, входящих в список x_i, задаются в подпрограмме-функции обычными средствами.

Оператор ENTRY является невыполняемым оператором, он не влияет на порядок выполнения других операторов и может размещаться внутри подпрограммы в любой ее точке. Этот оператор играет по отношению к функции f_i такую же роль, что и оператор

FUNCTION $f(x)$

по отношению к функции f.

В список формальных параметров оператора ENTRY могут входить формальные параметры оператора FUNCTION, формальные параметры других операторов ENTRY, а также другие идентификаторы. Никаких согласований по количеству, порядку следования, типу и длине между формальными параметрами операторов ENTRY и оператора FUNCTION не требуется. Важно лишь то, чтобы для одинаковых параметров во всех операторах входа были указаны одинаковые способы вызова (по имени или по значению).

Фактические параметры указателя вспомогательной функции должны быть согласованы по количеству, порядку следования, типу и длине с формальными параметрами соответствующего оператора ENTRY таким же образом, как фактические параметры

указателя основной функции с формальными параметрами оператора FUNCTION.

Имя вспомогательной функции внутри подпрограммы-функции должно получать значение всякий раз, когда происходит обращение к подпрограмме за вычислением этой функции. В момент выхода из подпрограммы-функции это значение присваивается указателю вспомогательной функции.

Приведем пример подпрограммы-функции с двумя входами:

```
FUNCTION F1(X,Y)
INTEGER F1,X,Y
F1=X**2+2*X*Y+Y**2
RETURN
ENTRY F2(X,Z,W)
REAL F2,Z,W
F2=2*X**3+Z**5+W
RETURN
END
```

В этом примере F1 является основной функцией, а F2 — вспомогательной. F1, X и Y относятся к целому типу, а F2, Z и W — к вещественному.

В подпрограмме-функции основная и вспомогательные функции считаются эквивалентными, т. е. их значения размещаются в памяти с одного и того же места. Если основная и вспомогательные функции относятся к одному типу и имеют одинаковые длины, то свойством эквивалентности можно воспользоваться для упрощения подпрограммы.

Рассмотрим такой пример: пусть даны две функции:

$$F1(x) = x - \frac{x^3}{3} + \frac{x^5}{5} - \frac{x^7}{7}$$

$$F2(x) = F1(x + 1.57)$$

Тогда подпрограмму вычисления этих функций можно представить в следующем виде:

```
FUNCTION F2(X)
X=X+1.57
ENTRY F1(X)
F1=X-X**3/3+X**5/5-X**7/7
RETURN
END
```

Так как F1 и F2 совмещены по памяти, то при вычислении значения величины F1 величина F2 получит то же самое значение. При обращении к подпрограмме-функции с именем F2 и заданным значением $X=X_0$ значение функции $F2(X_0)$ после выполнения подпрограммы-функции будет равно значению функции $F1(X_0+1.57)$.

Если в результате обращения к подпрограмме-функции некоторые из ее формальных параметров получили значения, то эти значения можно использовать при последующих обращениях к подпрограмме. Рассмотрим такой пример: пусть имеется основная программа

```
INTEGER F1,K1,K2
K1=F1(2,3)
K2=F2(4)
STOP
END
```

и подпрограмма-функция

```
FUNCTION F1(X,Y)
INTEGER F1,X,Y
F1=5*X+6*Y
RETURN
ENTRY F2(Z)
INTEGER F2,Z
F2=7*Z+8*Y+X
RETURN
END
```

При первом обращении к подпрограмме с помощью указателя функции F1(2,3) величина X получит значение, равное 2, а величина Y — значение, равное 3. При втором обращении к подпрограмме с помощью указателя функции F2(4) величина Z получит значение, равное 4, а в качестве значений X и Y будут браться их старые значения (2 и 3 соответственно).

Описанным выше приемом можно пользоваться только в том случае, когда основная программа, из которой осуществляется обращение, и подпрограмма-функция входят в один и тот же загрузочный сегмент.

Вопросы и упражнения

1. Какая функция подпрограммы-функции называется основной и какая — вспомогательной?

2. Могут ли вспомогательные функции иметь одинаковые имена?

3. Может ли подпрограмма-функция начать выполняться при обращении к ней не по указателю функции, а от выполняемого оператора, предшествующего оператору входа в данную подпрограмму-функцию?

4. К числу каких операторов относится оператор входа в подпрограмму вспомогательной функции?

5. Существуют ли какие-либо особые требования к формальным параметрам вспомогательного оператора входа в подпрограмму-функцию и существует ли какая-либо их зависимость от формальных параметров основного оператора входа?

6. Как размещаются в памяти значения основной и вспомогательных функций?

7. Определить значение переменной Z, которое она получит в результате выполнения следующей программы:

```
Основная программа      Подпрограмма-функция
REAL B                  REAL FUNCTION S(Y)
X=2.                    Y=Y+2*Y**2
Z=B(X)                  RETURN
STOP                    ENTRY B(T)
END                     REAL B
                        R=T+T**3-1
                        B=(R-SQRT(R))/T
                        RETURN
                        END
```

8. Составить подпрограмму-функцию для вычисления значений величин

$$y = x^3 - x^2 + x - 0{,}4 \quad \text{и} \quad z = \frac{y}{1 + \dfrac{y}{1+y}}$$

и программу для вычисления значений функции

$$f(x) = \begin{cases} y + 1, & \text{если } x < 0 \\ z^2 - 1, & \text{если } x \geqslant 0 \end{cases}$$

для всех значений $x_1, x_2, \ldots, x_{10}$, где $x_1 = -0{,}6$, $x_i = x_{i-1} + 0{,}37$; $i = 2, 3, \ldots, 10$.

В программе использовать подпрограмму-функцию, предназначенную для вычисления значений величин Y и Z.

4.2. Подпрограмма-процедура. Оператор процедуры

Подпрограмма-процедура имеет вид:

s_0
s_1
.
.
.
s_{n-1}
s_n

где s_0 — заголовок процедуры; s_i ($i = 1, 2, \ldots, n-1$) — последовательность выполняемых и невыполняемых операторов, не содержащая заголовков функций, заголовков процедур и операторов конца; s_n — оператор конца.

Последовательность операторов s_i должна удовлетворять тем же требованиям, что и последовательность операторов s_i основной программы.

4.2.1. Заголовок процедуры. *Заголовок процедуры* имеет вид:

SUBROUTINE p s

где p — имя (идентификатор) процедуры; s — либо пусто, либо список формальных параметров.

Список формальных параметров имеет вид:

$(y_1, y_2, \ldots, y_n)$

где y_i ($i = 1, 2, \ldots, n$) — либо формальные параметры, либо формальные параметры, заключенные в косые черточки.

В качестве формальных параметров могут выступать символы * (звездочка), а также идентификаторы, которые внутри подпрограммы используются как имена простых переменных, массивов, внешних функций и процедур. В косые черточки могут заключаться только те формальные параметры, которые в подпрограмме-процедуре используются в качестве имен простых переменных. Список формальных параметров может отсутствовать. В этом случае заголовок процедуры имеет вид:

SUBROUTINE p.

Примеры заголовков процедур:

1) SUBROUTINE P1(X,Y,Z,*,*)
2) SUBROUTINE Q2(A,/B/)
3) SUBROUTINE C3

Для задания типа, длины и размерности формальных параметров используются обычные средства (явные и неявные операторы типа, предварительное соглашение, операторы размеров).

Формальные параметры подпрограмм-процедур не могут иметь начальных значений и не могут включаться ни в операторы эквивалентности, ни в операторы общих областей.

Вопросы и упражнения

1. Что можно использовать в качестве формальных параметров в заголовке процедуры?

2. В каком случае используются косые черточки в списке формальных параметров?

3. Какие средства используются при описании свойств (типа, длины, размерности) формальных параметров?

4. Можно ли в подпрограмме-процедуре задать (хотя бы с помощью оператора явного описания типа) начальные значения?

5. Могут ли формальные параметры включаться в состав операторов эквивалентности и общих областей?

6. Может ли отсутствовать список формальных параметров в заголовке подпрограммы-продедуры?

4.2.2. Оператор процедуры. Выполнение подпрограммы-процедуры. *Оператор процедуры* имеет вид:

CALL *ps*

где p — имя (идентификатор) процедуры, s — либо пусто, либо список $(x_1, x_2, \ldots, x_n)$ фактических параметров x_i $(i = 1, 2, \ldots, n)$.

В качестве фактических параметров оператора процедуры могут использоваться константы, имена переменных, арифметические и логические выражения, имена массивов, внешних функций и подпрограмм-процедур. Кроме того, фактические параметры могут иметь форму $\&n$, где n — метка оператора.

Оператор процедуры служит для обращения к подпрограмме-процедуре с именем p.

Подпрограмма-процедура выполняется при обращении к ней из другого программного модуля с помощью оператора процедуры.

Фактические параметры оператора процедуры должны удовлетворять следующим требованиям:

1. Количество фактических параметров оператора процедуры должно совпадать с количеством формальных параметров подпрограммы.

2. Тип и длина фактических параметров должны совпадать с типом и длиной соответствующих формальных параметров. Соот-

ветствие между параметрами устанавливается по порядку их следования в списках параметров (первому формальному параметру соответствует первый фактический, второму формальному параметру — второй фактический и т. д.).

3. Если формальный параметр — имя функции, то соответствующий фактический параметр должен быть именем внешней функции.

4. Если формальный параметр — имя процедуры, то соответствующий фактический параметр должен быть именем процедуры.

5. Если формальный параметр — имя массива, то в качестве соответствующего фактического параметра может указываться либо имя массива, либо переменная с индексом (элемент массива).

6. Если формальный параметр — имя переменной, которая в подпрограмме не получает новых значений, то в качестве соответствующего фактического параметра могут указываться константы, переменные и выражения. Если же формальный параметр получает новые значения, то соответствующим фактическим параметром может быть только переменная.

7. Если формальным параметром подпрограммы-процедуры является имя переменной или имя массива, то в качестве соответствующего фактического параметра оператора процедуры может использоваться и символьная константа. Эта константа размещается в оперативной памяти с границы двойного слова и занимает столько байтов, какова длина этой константы. Необходимо следить за тем, чтобы длина этой константы соответствовала длине формального параметра.

8. Если формальный параметр — символ * (звездочка), то соответствующий фактический параметр должен иметь вид: *&n*, где *n* — метка оператора.

Имя подпрограммы-процедуры не имеет типа и внутри подпрограммы-процедуры не должно использоваться нигде, кроме как в заголовке процедуры.

При обращении к подпрограмме-процедуре фактические параметры оператора процедуры могут вызываться в подпрограмму как по имени, так и по значению.

Оба эти способа рассмотрены при описании подпрограммы-функции.

Формальные массивы могут иметь регулируемые размеры. Особенности использования таких массивов также рассмотрены при описании подпрограммы-функции.

Выполнение подпрограммы-процедуры начинается с первого выполняемого оператора и должно завершаться оператором воз-

врата. В подпрограммах-процедурах используются операторы возврата двух типов: обычный оператор возврата RETURN и оператор возврата по метке, который имеет вид:

RETURN *i*

Здесь *i* — либо простая переменная целого типа стандартной длины, либо целая константа без знака.

Оператор RETURN передает управление в вызывающий программный модуль на оператор, следующий за оператором обращения к подпрограмме.

Оператор RETURN *i* можно использовать только в том случае, если список формальных параметров содержит символ * (звездочка). Он передает управление в вызывающий программный модуль на оператор, метка которого содержится в списке фактических параметров оператора процедуры и соответствует *i*-й звездочке (если перечислять их слева направо) в списке формальных параметров подпрограммы-процедуры.

В том случае, если подпрограмма-процедура не содержит списка формальных параметров, то обмен данными с ней можно осуществлять через общие области. Такой способ используется в приведенной ниже программе.

```
Основная программа          Подпрограмма-процедура
DIMENSION A(10)             SUBROUTINE P1
COMMON X,Y,A                DIMENSION B(10)
Y=—3.7                      COMMON P,Q,B
DO 1 K=1,10                 P=0
1 A(K)=K+0.5                DO 1 J=1,10
CALL P1                   1 P=P+B(J)
STOP                        P=P*Q**2+Q/3
END                         RETURN
                            END
```

После вычисления значения величины Y и массива A происходит обращение к подпрограмме-процедуре с именем P1. Операторы общих областей основной программы и подпрограммы указывают на совмещение значений X и P, Y и Q, A(1) и B(1), A(2) и B(2), . . ., A(10) и B(10). Таким образом, будет вычислено значение величины P, а следовательно, и величины X.

Одна подпрограмма-процедура может содержать несколько операторов возврата.

Рассмотрим пример: пусть имеются оператор процедуры

CALL Q(A, B,&50, &70, &35)

и подпрограмма-процедура

```
SUBROUTINE Q(X,Y,*,*,*)
. . . . . .
RETURN 2
END
```

Оператор RETURN 2, содержащийся в подпрограмме-процедуре, передаст управление оператору с меткой 70, который содержится в том же программном модуле, что и оператор

```
CALL Q(A,B,&50,&70,&35)
```

с помощью которого осуществлялось обращение к подпрограмме.

Подпрограмма-процедура не должна содержать обращений к самой себе как непосредственно, так и через другие подпрограммы. Имена процедур, входящих в списки фактических параметров операторов процедур, должны быть перечислены в операторе внешних подпрограмм, входящем в состав того программного модуля, который содержит эти указатели функций и операторы процедур.

Примеры подпрограмм-процедур:

1)
```
SUBROUTINE P1(A,B)
INTEGER A,B
B=3*A**2—6
RETURN
END
```

2)
```
SUBROUTINE RAD(/A4/,/B4/,*,*)
B4=A4**4+A4**3
IF (B4)1,2,3
1 RETURN
2 RETURN 1
3 RETURN 2
END
```

Вопросы и упражнения

1. Для каких целей используется оператор процедуры?

2. Что может использоваться в качестве фактических параметров оператора процедуры?

3. Что определяет фактический параметр &10?

4. Перечислить требования, которым должны отвечать фактические параметры оператора процедуры.

5. Указать, какие способы существуют для вызова фактических параметров при обращении к подпрограмме-процедуре.

6. На что указывает целая константа без знака в операторе RETURN 3?

7. Как осуществляется обмен данными с подпрограммой-процедурой, не содержащей списка формальных параметров в заголовке подпрограммы?

8. Какие требования предъявляются к именам процедур, которые используются в операторе процедуры в числе фактических параметров?

4.2.3. Примеры программ, содержащих подпрограммы-процедуры.

```
1) Основная программа          Подпрограмма-процедура
   DIMENSION A(8)              SUBROUTINE P1(B)
   DO 2 I=1,8                  DIMENSION B(5)
 2 A(I)=0.0                    DO 1 I=2,5
   CALL P1(A)                1 B(I)=B(I-1)+1.0
   STOP                        RETURN
   END                         END
```

Особенность этой программы состоит в том, что массив, входящий в состав фактических параметров оператора процедуры, имеет элементов больше, чем соответствующий массив, входящий в состав формальных параметров заголовка процедуры, что в соответствии с требованием, предъявляемым к фактическим параметрам, является допустимым.

В результате выполнения этой программы элементы массива A примут значения A(1)=A(6)=A(7)=A(8)=0.0; A(2)=1.0; A(3)=2.0; A(4)=3.0; A(5)=4.0

```
2) Основная программа          Подпрограмма-процедура
   DIMENSION A(3,3)            SUBROUTINE P1(A)
   DO 2 I=1,3                  DIMENSION A(5)
   DO 2 J=1,3                  A(1)=1.0
 2 A(I,J)=0.0                  DO 1 I=2,5
   CALL P1(A)                1 A(I)=A(I-1)+1.0
   STOP                        RETURN
   END                         END
```

Особенность этой программы заключается в том, что массив, входящий в состав фактических параметров оператора процедуры, описан в основной программе как двумерный массив, тогда как соответствующий ему массив, входящий в состав формальных параметров, описан в подпрограмме как одномерный. Соответствие

элементов этих массивов устанавливается по их порядковым номерам, т. е. элементу A(1,1) основной программы соответствует элемент A(1) процедуры; элементу A(2,1) — соответствует элемент A(2); элементу A(3,1) — элемент A(3); элементу A(1,2) — элемент A(4) и, наконец, элементу A(2,2) — элемент A(5).

В результате выполнения этой программы элементы массива A примут значения: A(1,1)=1.0; A(2,1)=2.0; A(3,1)=3.0; A(1,2)=4.0; A(2,2)=5.0; A(2,3)=A(1,3)=A(3,2)=A(3,3)=0.0

```
3) Основная программа        Подпрограмма-процедура
   DIMENSION A(4,4)          SUBROUTINE MEN(B,X,Y)
   DO 1 K=1,4                INTEGER X,Y
   DO 1 J=1,4                DIMENSION B(X,Y)
 1 A(K,J)=0.0                DO 1 I=1,X
   CALL MEN(A,3,2)           DO 1 J=1,Y
   STOP                    1 B(I,J)=1.0
   END                       RETURN
                             END
```

Заголовок подпрограммы-процедуры в списке формальных параметров содержит регулируемые размеры двумерного массива B.

В результате выполнения данной программы элементы массива A примут значения: A(1,1)=A(1,2)=A(2,1)=A(2,2)=A(3,1)=A(4,1)=1.0; A(1,3)=A(1,4)=A(2,3)=A(2,4)=A(3,3)=A(3,4)=A(3,2)=A(4,2)=A(4,3)=A(4,4)=0.0

```
4) Основная программа    Подпрограммы-процедуры
INTEGER X,Z              SUBROUTINE T1(A,B,C)
EXTERNAL C1              INTEGER A,B
X=2                      B=A**2
CALL T1(X,Z,C1)          CALL C(B)
STOP                     RETURN
END                      END
                         SUBROUTINE C1(X)
                         INTEGER X
                         X=4*X
                         RETURN
                         END
```

Особенность программы состоит в том, что в состав фактических параметров оператора процедуры входит имя процедуры C1. Именно поэтому в основную программу, в которой содержится этот оператор процедуры, включен оператор внешних подпрограмм, содержащий имя C1 подпрограммы-процедуры.

В результате выполнения программы величина Z примет значение, равное числу 16.

Вопросы и упражнения

1. В каком случае применяется оператор внешних подпрограмм и где должен размещаться этот оператор?

2. Какие размеры массивов называются регулируемыми?

3. Определить значения элементов массива A, которые они получат в результате выполнения следующей программы:

```
  Основная программа          Подпрограмма-процедура
  DIMENSION A(4,4)            SUBROUTINE M(B,X,Y)
  DO 1 K=1,4                  INTEGER X,Y
  DO 1 J=1,4                  DIMENSION B(X,Y)
1 A(K,J)=0.0                  DO 1 I=2,X
  CALL M(A,3,2)               DO 1 J=2,Y
  STOP                      1 B(I,J)=B(I-1,J-1)+0.5
  END                         RETURN
                              END
```

4. Определить значения переменных X и Y, которые они получат в результате выполнения следующей программы:

```
  Основная программа          Подпрограмма-процедура
  X=2.0                       SUBROUTINE R(A,B,*,*)
  Y=2.5                       A=A+1
  CALL R(X,Y,&2,&3)           B=B+1
2 X=X**2                      IF (A-B)1,1,2
  GO TO 6                   1 RETURN 1
3 Y=Y**2                    2 RETURN 2
6 CALL R(X,Y,&4,&5)           END
4 X=2*X
  GO TO 7
5 Y=2*Y
7 STOP
  END
```

4.2.4. Кратный вход в подпрограмму-процедуру. Подпрограмма-процедура может представлять собой объединение нескольких подпрограмм. Одна из них считается основной, а остальные вспомогательными.

Имя основной подпрограммы указывается в заголовке процедуры, который является основным оператором входа в подпрограмму-процедуру. Обращение к основной подпрограмме мы уже рассматривали.

Обращение к вспомогательной подпрограмме осуществляется с помощью оператора процедуры CALL $p_i s_i$, где p — имя i-й вспомогательной подпрограммы, s_i либо пусто, либо список $(x_1, x_2, \ldots, x_n)$ фактических параметров x_j $(j = 1, 2, \ldots, n)$. В этом случае выполнение подпрограммы-процедуры начинается с первого выполняемого оператора, следующего за вспомогательным оператором входа, относящимся к подпрограмме p_i.

Вспомогательный оператор входа является невыполняемым оператором и не влияет на порядок выполнения других операторов. Он имеет вид:

ENTRY $p_i s$

где p_i — имя вспомогательной подпрограммы, s — либо пусто, либо список $(y_1, y_2, \ldots, y_n)$ формальных параметров y_i $(i = 1, 2, \ldots, n)$.

Тип и длина формальных параметров должны быть заданы внутри подпрограммы-процедуры обычными средствами.

В список формальных параметров оператора ENTRY могут входить формальные параметры других операторов ENTRY, формальные параметры оператора SUBROUTINE, а также другие идентификаторы и символ *.

Никаких согласований по количеству, порядку следования, типу и длине между формальными параметрами операторов ENTRY и оператора SUBROUTINE не требуется. Важно лишь то, чтобы для одинаковых параметров во всех операторах входа были указаны одинаковые способы вызова (по имени или по значению).

При обращении к вспомогательной подпрограмме фактические параметры оператора CALL должны быть согласованы по количеству, порядку следования, типам и длинам с формальными параметрами соответствующего оператора ENTRY.

Выход из подпрограммы-процедуры, содержащей несколько операторов входа, осуществляется в соответствии с оператором процедуры и оператором входа, посредством которых осуществлялось обращение к подпрограмме-процедуре.

Рассмотрим примеры.

1. Пусть имеется основная программа и подпрограмма-процедура

```
Основная программа          Подпрограмма-процедура
INTEGER A,B,E               SUBROUTINE P1(X,Y)
REAL C,D                    INTEGER X,Y
A=2                         Y=3*X**3+35*X**2—83
CALL P1(A,B)              1 RETURN
C=—1.3                      ENTRY P2(K,X,Z)
D=3.5                       REAL K,Z
CALL P2(C,E,D)              X=K**3+Z
STOP                        GO TO 1
END                         END
```

Выполнение основной программы начинается с выполнения оператора A=2. Затем выполняется оператор процедуры CALL P1(A,B). В результате его выполнения вызывается подпрограмма с именем P1; данные из основной программы передаются в подпрограмму по способу, который называется вызовом фактического параметра по значению; вычисляется значение величины Y и осуществляется возврат в основную программу к оператору C=—1.3. Вычисляются значения величин C и D и управление получает оператор процедуры CALL P2(C,E,D), при выполнении которого вызывается подпрограмма с именем P2. Осуществляется передача фактических параметров C и D в подпрограмму, производится вычисление величины X и происходит возврат в основную программу к оператору STOP.

2. Пусть имеется основная программа и подпрограмма-процедура:

```
  Основная программа        Подпрограмма-процедура
  READ (15)T                SUBROUTINE P3(T1,X1,Y1,*)
  IF (T)1,2,2               X1=3.5+T1*7.8+T1**3
1 CALL P3(T,X,Y,&3)         ENTRY P4(T1,Y1,*)
2 CALL P4(T,Y,&4)           RETURN 1
3 WRITE (16)T,X,Y           END
  GO TO 5
4 WRITE (16)T,Y
5 STOP
  END
```

Забегая несколько вперед, укажем, что в основной программе содержатся оператор ввода значения величины T (READ (15)T) и два оператора вывода: первый оператор WRITE (16)T,X,Y осуществляет вывод значений величин T, X и Y; второй WRITE (16)T,Y — величин T и Y.

Выполнение основной программы начинается с ввода значения величины T и в зависимости от этого значения по оператору IF (T)1,2,2 управление передается оператору с меткой 1, т. е. оператору процедуры CALL P3(T,X,Y,&3) или оператору с меткой 2, т. е. оператору процедуры CALL P4(T,Y,&4).

Если обращение к подпрограмме из основной программы будет осуществляться с помощью оператора CALL P3(T,X,Y,&3), то после выполнения подпрограммы управление будет передано оператору, помеченному меткой 3. Если же обращение к подпрограмме будет осуществляться с помощью оператора CALL P4 (T,Y, &4) то после выполнения подпрограммы управление будет передано оператору, помеченному меткой 4.

Если в результате обращения к подпрограмме-процедуре некоторые из ее формальных параметров получили значения, то эти значения можно использовать при последующих обращениях к подпрограмме. Способы и условия использования аналогичны тем, которые рассмотрены нами при описании подпрограммы-функции.

Вопросы и упражнения

1. Обязательно ли выполнение подпрограммы процедуры начинается с первого выполняемого оператора этой подпрограммы?

2. Какая подпрограмма называется основной и какая вспомогательной?

3. Перечислить требования, которые предъявляются к формальным параметрам вспомогательной процедуры и к фактическим параметрам оператора процедуры, указывающего на обращение к этой процедуре.

4. Пусть задана подпрограмма-процедура

```
SUBROUTINE A(X,Y)
(последовательность операторов)
RETURN
END
```

предназначенная для вычисления величины $y = f(x)$.

Составить программу вычисления величины y в точках $x = -3.43,\ 2.72,\ 9.99,\ 15.16,\ -3,99,\ 88.8$, используя приведенную выше подпрограмму.

5. Пусть задана подпрограмма-процедура

```
SUBROUTINE SUM(A,N,B,*)
DIMENSION A(N)
```

```
      B=0.0
      DO 1 K=1,N
    1 B=B+A(K)
      RETURN 1
      END
```

предназначенная для вычисления величины $B = \sum_{i=1}^{N} a_i$.

Составить программу вычисления величины $D = \sum_{i=1}^{20} p_i$, используя приведенную выше подпрограмму, если $p_1 = -0.36$, $p_2 = 17.64$, $p_3 = 2.07$, $p_4 = p_5 = p_6 = p_7 = p_8 = 1.2$, $p_9 = -3.8$, $p_{10} = p_{11} = p_{12} = p_{13} = p_{14} = p_{15} = -64.2$, $p_{16} = p_{17} = p_{18} = p_{19} = p_{20} = -1.0$

6. Пусть задана подпрограмма-процедура

```
      SUBROUTINE P(A,X,Y,N)
      DIMENSION A(N)
      Y=A(1)
      DO 1 J=2,N
    1 Y=Y*X+A(J)
      RETURN
      ENTRY P1(A,X,Y,N)
      Y=0.0
      DO 2 K=1,N
    2 Y=Y+A(K)
      Y=Y+X-1.3
      RETURN
      END
```

предназначенная для вычисления величины

$y = a_1x^N + a_2x^{N-1} + \ldots + a_Nx + a_{N+1}$ и величины $y = \sum_{i=1}^{N+1} a_i + x - 1.3$.

Составить программу вычисления значений функции

$$f(x) = \begin{cases} a_1x^n + a_2x^{n-1} + \ldots + a_nx + a_{n+1}, & \text{если } x < 0 \\ x - 1.3 + \sum_{i=1}^{n+1} a_i, & \text{если } x \geqslant 0 \end{cases}$$

для всех значений $x = -1.6, 2.3, -16.2, 3.7, 3.8, -4,5$ и вектора $a\ (a_1, a_2, \ldots, a_{10})$, в котором $a_1 = a_2 = a_3 = \ldots = a_{10} = 1.0$

4.2.5. Подпрограмма начальных значений. Подпрограммы начальных значений используются для задания начальных значений величинам, входящим в именованные общие области. Они имеют вид:

s_0
s_1
.
.
.
s_n

где s_0 — заголовок подпрограммы начальных значений; s_i ($i = 1, 2, \ldots, n-1$) — последовательность невыполняемых операторов; s_n — оператор конца.

Заголовок подпрограммы начальных значений имеет вид:

```
BLOCK DATA
```

Последовательность операторов s_i может содержать следующие операторы: неявный оператор типа (только один), операторы размеров, операторы общих областей, явные операторы типа.

Если неявный оператор типа в подпрограмме используется, то он должен следовать непосредственно за заголовком подпрограммы. Операторы, в которых указываются начальные значения величин, должны размещаться после операторов общих областей, содержащих имена этих величин. Начальные значения могут указываться не для всех величин, входящих в некоторую общую область.

Пример подпрограммы начальных значений:

```
BLOCK DATA
COMMON /R1/A,B,C,/R2/X,Y,Z
REAL A/3.5/,C*8/7.44D15/
COMPLEX Y(15)/15*(2.4,3.769)/
END
```

Начальные значения величин, входящих в состав некоторой общей области, могут указываться только в одной подпрограмме начальных значений, в то время как программа может содержать несколько подпрограмм начальных значений.

Размер некоторой общей области, описанной в подпрограмме начальных значений, должен быть не меньше размеров одноименных общих областей, описанных в других программных единицах.

Глава 5

ОРГАНИЗАЦИЯ НАБОРОВ ДАННЫХ

5.1. Наборы данных

Операционная система ОС ЕС ЭВМ обеспечивает возможность использования в процессе решения задач большого количества разнотипных внешних запоминающих устройств, а также оконечных устройств для ввода и вывода значений величин (данных). Данные на этих устройствах принято размещать группами (наборами), которым присваиваются определенные имена.

5.1.1. Организация данных в наборах. Наборы данных делятся на блоки. Блок может содержать одну или несколько физических записей, под каждую из которых отводится определенное количество позиций, называемое длиной физической записи.

Позиция — это элемент носителя информации, обеспечивающий хранение элементарной порции данных. Для внешних носителей информации такими элементарными порциями данных являются отдельные символы, произвольные восьмиразрядные двоичные коды и др. Так, например, позиции на магнитной ленте и магнитном диске служат для хранения произвольных восьмиразрядных кодов, а также символов во внутримашинном коде ДКОИ. Позиции на перфокарте служат для хранения символов в перфокарточном коде ЕС ЭВМ, позиции на бумажной ленте — для хранения символов в графическом виде и т. д. Значение величины занимает в наборе данных поле, состоящее, как правило, из нескольких позиций.

Засылка данных в наборы и их чтение из наборов в фортран-программах осуществляется порциями, которые принято называть логическими записями. Логическая запись характеризуется своей длиной, под которой понимается количество позиций набора, необходимых для размещения данных, составляющих логическую запись. Длина логической записи может совпадать с длиной физической записи набора, быть больше или меньше ее. Допустимая между

ними зависимость определяется способом организации набора, а также форматом его физических записей.

Большинство характеристик наборов данных (размеры блоков, размеры физических записей, форматы физических записей, шифры устройств, на которых размещены наборы данных и т. д.) указываются в управляющих операторах задания; некоторые из них указываются в операторах ввода-вывода.

Работа с набором данных начинается с его открытия, в процессе чего над набором выполняется ряд подготовительных действий. Наборы данных могут открываться как входные, как выходные и как обновляемые. К входным наборам можно обращаться только за вводом данных, к выходным — только за выводом данных. Если же набор открыт как обновляемый, то к нему можно обращаться как за вводом, так и за выводом данных.

Работа с набором данных завершается процедурой закрытия, в процессе чего над набором выполняется ряд завершающих действий. Открытие наборов данных осуществляется операторами ввода-вывода, а закрытие — либо операторами ввода-вывода, либо операционной системой после завершения выполнения программы. Обращение к наборам данных осуществляется по их ссылочным номерам. *Ссылочные номера* — это условные имена наборов. Соответствие между ссылочными номерами и истинными именами наборов данных указывается в управляющих операторах задания.

В языке фортран используются два типа наборов данных: с последовательной организацией и с прямой организацией.

Наборы с последовательной организацией обеспечивают перебор записей только в последовательном порядке. Это означает, что ввод или вывод данных всегда начинается с первой записи и в дальнейшем обеспечивается переход только к последующей или только к предыдущей записи.

Наборы с прямой организацией обеспечивают перебор записей в произвольном порядке.

Наборы с последовательной организацией могут создаваться на любых внешних запоминающих устройствах. Наборы с прямой организацией могут создаваться только на устройствах прямого доступа (на магнитных дисках и магнитных барабанах). Ввод и вывод осуществляется специальными модулями операционной системы, входящими в систему ввода-вывода информации. Система ввода-вывода информации позволяет составлять программы, не учитывающие особенности конкретных устройств ввода-вывода. Однако особенности типов устройств ввода-вывода учитывать необходимо. Так, например, на магнитных лентах и перфокартах

можно создавать наборы только последовательного доступа, а на магнитных дисках — как последовательного, так и прямого доступа. Имеются ограничения на размеры блоков и т. д. Особенности работы с различными типами устройств ввода-вывода будут описаны ниже.

Вопросы и упражнения

1. Дать определение набора данных и описать его структуру.

2. Описать особенности использования входных, выходных и модифицируемых наборов данных.

3. Что такое ссылочный номер набора данных и каково его назначение?

4. Назвать типы наборов данных, используемых в языке фортран.

5. Чем отличается набор данных с прямой организацией от набора данных с последовательной организацией?

6. На каких устройствах могут размещаться наборы данных с прямой организацией?

7. Какими процедурами начинается и заканчивается работа с наборами данных?

5.1.2. Буферы ввода-вывода. С каждым набором данных связываются области оперативной памяти, называемые буферами ввода-вывода. *Буферы ввода-вывода* служат промежуточной памятью для работы операторов ввода-вывода. Так как обмен данными с внешними носителями осуществляется поблочно, то размер каждого буфера берется таким, чтобы он мог разместить содержимое самого большого блока того набора, с которым он связан.

Буферы выделяются набору в момент его открытия и отбираются у него в момент его закрытия. Так как выделение буферов осуществляется из той области оперативной памяти, которая отводится операционной системой для данной программы, то в целях экономии оперативной памяти нужно стремиться к быстрейшему освобождению буферов (к быстрейшему закрытию использованных наборов данных).

Операционная система позволяет каждому набору данных выделять несколько буферов. Использование нескольких буферов приводит к убыстрению процесса ввода-вывода. В программах на языке фортран операторы ввода-вывода могут использовать один или два буфера на каждый набор. Количество используемых буферов указывается в специальных управляющих операторах задания. Если это не сделать, то каждому набору данных будет выделено по два буфера.

При вводе содержимое блока помещается в буфер, а затем пересылается в те участки оперативной памяти, которые отведены под значения вводимых величин.

При выводе значения выводимых величин вначале накапливаются в буфере, образуя блок, а затем пересылаются в набор.

Вопросы и упражнения

1. Какие операторы языка фортран используют буферы ввода-вывода?

2. Сколько буферов ввода-вывода может быть выделено одному набору данных в программе, написанной на языке фортран?

3. Как определяется размер буфера?

4. В какие моменты выполнения программы осуществляется выделение и отбор буферов?

5.1.3. Форматы блоков данных. Блоки, образующие тот или иной набор данных, должны относиться к одному из следующих типов: фиксированной длины, переменной длины или неопределенной длины.

Блоки фиксированной длины, входящие в состав некоторого набора данных, должны состоять из одинакового количества физических записей, причем все эти записи должны иметь одинаковую длину (одинаковое количество позиций). Исключением может быть последний блок набора, который может содержать меньшее количество записей.

Структуру блоков принято характеризовать *форматами*. Блоки фиксированной длины, состоящие из одной физической записи, имеют формат F (формат фиксированной длины), а блоки фиксированной длины, состоящие из нескольких физических записей — формат FB (формат фиксированной длины, блокированный).

В блок фиксированной длины никакой служебной информации не засылается, и все его позиции предназначаются только для хранения данных. Фрагменты наборов данных с блоками фиксированной длины приведены на рис. 5.1 и рис. 5.2. На этих рисунках приняты следующие обозначения: D — поле данных, Z — поле физической записи, B — поле блока, P — поле промежутка между блоками.

Форматы блоков, а также длины блоков и физических записей указываются в специальных управляющих операторах задания (оператор DD), причем длины блоков должны быть кратными длинам физических записей.

Блоки неопределенной длины, входящие в состав некоторого набора данных, могут включать в свой состав только по одной физи-

ческой записи. Эти записи могут иметь разную длину. Блоки неопределенной длины имеют формат неопределенной длины, который обозначается буквой U. В эти блоки никакой служебной информации не заносится, и все их позиции предназначаются только для хранения данных.

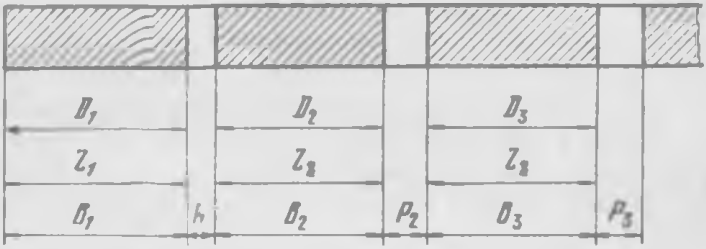


Рис. 5. 1. Фрагмент набора данных с блоками, имеющими форматы F и U.

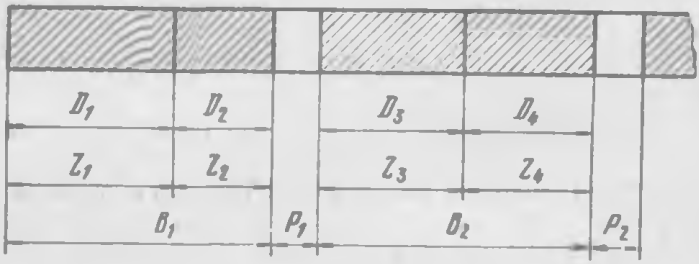


Рис. 5. 2. Фрагмент набора данных с блоками, имеющими формат FB.

Фрагмент набора данных с блоками неопределенной длины приведен на рис. 5.1.

Формат и максимальная длина блоков неопределенной длины указываются в управляющих операторах задания. Фактические длины блоков определяются в процессе выполнения операторов вывода в зависимости от длин логических записей. Однако они не превышают указанных в задании максимальных длин.

Возможные соотношения длин логических и физических записей для наборов данных с форматами блоков F, FB и U зависят от многих обстоятельств и будут рассмотрены в процессе описания операторов ввода-вывода.

Блоки переменной длины, входящие в состав некоторого набора данных, могут состоять из разного количества физических записей, причем все эти записи могут иметь разную длину.

Блоки переменной длины, допускающие включение в свой состав только по одной физической записи, имеют формат V (формат переменной длины), а блоки переменной длины, допускающие включение в свой состав по несколько физических записей, имеют формат VB (формат переменной длины, блокированный).

Характерной особенностью блоков переменной длины является то обстоятельство, что они содержат информацию о своей длине и

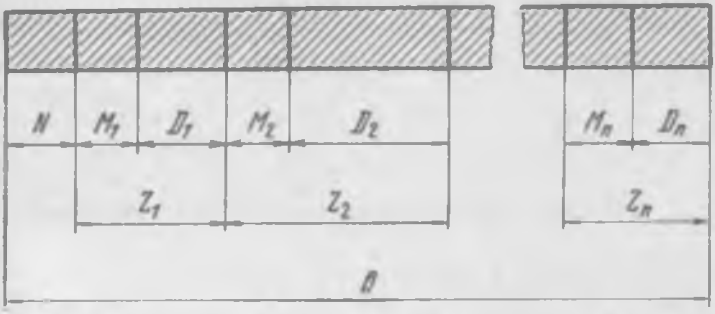


Рис. 5. 3. Структура блока данных, имеющего формат VB.

о длине физических записей, входящих в их состав. Эта информация помещается в блоки в процессе выполнения операторов вывода. Структура блока данных переменной длины, имеющего формат VB, приведена на рис. 5.3, а фрагменты наборов данных с блоками переменной длины приведены на рис. 5.4 и рис. 5.5. На этих рисунках приняты следующие обозначения: N — четырехбайтовое служебное поле, в котором содержится длина блока, M — четырехбайтовое служебное поле, в котором содержится длина физической записи и другая информация, D — поле данных, Z — поле физической записи, B — поле блока, P — поле промежутка между блоками данных.

Длину блока, имеющего формат VB, можно вычислить по формуле:

$$l_B = 4 + 4 \cdot n + \sum_{1}^{n} l_i$$

где l_B — длина блока в байтах, n — количество физических записей, l_i — длина i-го поля данных в байтах.

Форматы блоков, а также максимальные длины блоков и физических записей указываются в управляющих операторах задания.

В фортран-программах разрешается использовать блоки переменной длины с возможностью сегментирования логических записей,

форматы которых обозначаются через VS и VBS. При использовании таких блоков на длины логических записей не накладывается никаких ограничений. Они могут быть меньше максимальных длин блоков, равны им и превосходить их. В первых двух случаях каждая логическая запись помещается в отдельный блок. В последнем

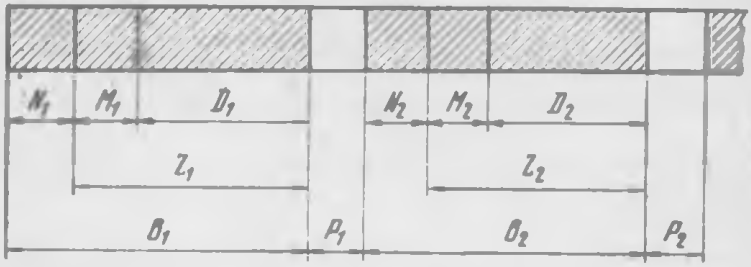


Рис. 5. 4. Фрагмент набора данных с блоками, имеющими формат V.

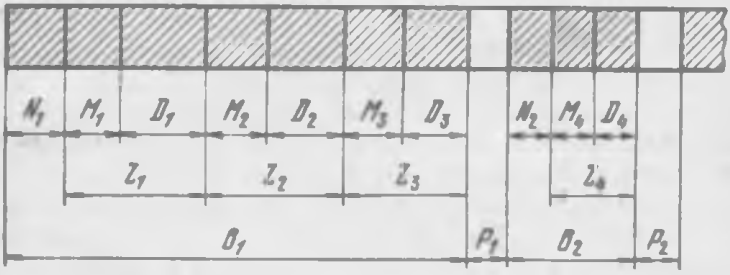


Рис. 5. 5. Фрагмент набора данных с блоками, имеющими формат VB.

случае логическая запись разбивается на сегменты, каждый из которых помещается в отдельный блок. При этом все блоки, кроме последнего, будут иметь максимально возможную длину. В последний блок будет помещен последний сегмент логической записи, и его длина может оказаться меньше максимально возможной.

Если блоки имеют формат VS, то следующая логическая запись будет размещаться со следующего блока. Если же блоки имеют формат VBS и длина последнего блока оказалась меньше максимально возможной, то первый сегмент следующей логической записи будет помещен в этот же блок (следом за последним сегмен-

том предыдущей логической записи), и его длина будет скорректирована.

Пример такого блока приведен на рис. 5.6. На этом рисунке приняты следующие обозначения: N — четырехбайтовое поле, содержащее длину блока; M — четырехбайтовое служебное поле,

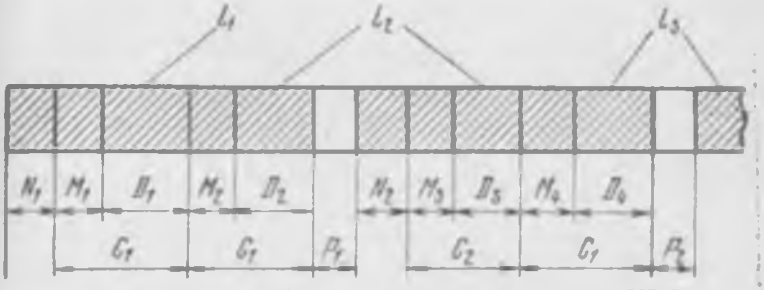


Рис. 5. 6. Пример блока данных с форматом VBS.

содержащее длину сегмента и описание сегмента; D — поле данных; C — поле сегмента; L — совокупность полей, составляющих логическую запись. *Описание сегмента* — это величина, значение которой зависит от того, является ли данный сегмент первым в логической записи, последним или промежуточным.

Вопросы и упражнения

1. Какие форматы блоков допускаются в наборах данных, используемых в программах на языке фортран?

2. В каких случаях блоки набора данных содержат, кроме данных, служебную информацию и сколько позиций она занимает?

3. Каким образом размещаются данные логической записи в наборе, если формат блока этого набора VS, размер логической записи превышает максимально возможный размер блока?

5.1.4. Формы представления данных. Данные в наборах и в оперативной памяти могут представляться в машинных формах и в символьных формах.

Машинные формы представления данных являются «рабочими» формами. Все арифметические и логические операции, предусмотренные в программе, могут выполняться над данными, представленными только в машинных формах.

Символьные формы, как наиболее удобные для человека, используются, в основном, для представления вводимых и выводимых данных в виде десятичных чисел, текстов и т. д.

Машинные формы данных в зависимости от их типов и длин приведены в табл. 5.1. Символьные формы данных описываются в

Таблица 5.1. Машинные формы представления данных

Тип данных	Длина	Машинная форма представления
Целый	Стандартная	Четырехбайтовое двоичное число с фиксированной точкой.
Целый	Нестандартная	Двухбайтовое двоичное число с фиксированной точкой.
Вещественный	Стандартная	Четырехбайтовое двоичное число с плавающей точкой.
Вещественный	Нестандартная	Восьмибайтовое двоичное число с плавающей точкой.
Комплексный	Стандартная	Два четырехбайтовых двоичных числа с плавающей точкой.
Комплексный	Нестандартная	Два восьмибайтовых двоичных числа с плавающей точкой.
Логический	Стандартная	Истина — число 1, представленное в виде четырехбайтового двоичного числа с фиксированной точкой. Ложь — число 0, представленное в виде четырехбайтового двоичного числа с фиксированной точкой.
Логический	Нестандартная	Истина — число 1, представленное в виде однобайтового двоичного числа с фиксированной точкой. Ложь — число 0, представленное в виде однобайтового двоичного числа с фиксированной точкой.
Символьный		Каждый символ представляется в виде восьмиразрядного двоичного кода.

программах с помощью форматов. Перевод данных из символьных форм в машинные и обратный перевод осуществляются операторами форматного ввода-вывода.

Вопросы и упражнения

1. Назвать формы представления данных в наборах.
2. Назвать машинные формы представления числовых данных.
3. Пусть имеется набор данных со следующими характеристиками: формат блоков — F, длина блоков — 500 байтов.

Определить минимальное количество блоков, необходимое для размещения в них значений 1400 величин целого типа стандартной длины.

4. Пусть имеется набор данных со следующими характеристиками: формат блоков — V, максимальная длина блоков — 500 байтов.

Определить минимальное количество блоков, необходимое для размещения в них 1500 величин вещественного типа стандартной длины.

5. Пусть имеется набор данных со следующими характеристиками: формат блоков — VB, максимальная длина блока — 804 байта, максимальная длина физической записи — 200 байтов.

Определить максимальную длину (в байтах) логической записи, которую можно разместить в одном блоке набора.

6. Пусть имеется логическая запись длиной 400 байтов, которую нужно разместить в одном блоке набора данных, имеющем следующие характеристики: формат блока — VB, количество физических записей в блоке — 4. Все физические записи имеют одинаковую длину.

Определить длину блока и длину физической записи в блоке.

5.2. Списки вводимых и выводимых величин

Списки вводимых и выводимых величин являются составными частями операторов ввода и вывода данных. Они представляют собой перечни переменных и массивов, значения которых берутся из наборов данных или помещаются в наборы данных. Эти списки имеют вид:

$$a_1, a_2, \ldots, a_n$$

где a_i ($i = 1, 2, \ldots, n$) — элементы списка. В качестве элементов списка могут использоваться имена переменных (простых и с индексами), имена массивов и циклические элементы.

Имя переменной целого, вещественного или логического типов указывает на ввод или вывод одного значения, имя переменной комплексного типа — на ввод или вывод двух значений (значения действительной части и значения мнимой части), а имя массива — на ввод или вывод значений всех его элементов.

Циклический элемент имеет вид:

$$(b_1, b_2, \ldots, b_k, c)$$

где b_j ($j = 1, 2, \ldots, k$) — элементы списка (имена переменных, имена массивов и циклические элементы); c — заголовок цикла.

Заголовок цикла в циклическом элементе списка играет ту же роль, что и в операторе цикла. Он определяет повторение элементов списка, входящих в состав циклического элемента. Ниже приводятся примеры циклических элементов списка и эквивалентные им фрагменты списка, не содержащие циклических элементов.

Циклический элемент	Эквивалентный фрагмент списка
(A(K),B(K),K=1,7,2)	A(1),B(1),A(3),B(3),A(5), B(5),A(7),B(7)
(A(I,J+1),B(J),J=3,9,3)	A(I,4),B(3),A(I,7),B(6), A(I,10),B(9)
(A,B(1,N),N=2,4)	A,B(1,2),A,B(1,3),A,B(1,4)
(A(K),(B(K,J),J=7,8), K=3,4)	A(3),B(3,7),B(3,8),A(4), B(4,7),B(4,8)

Вопросы и упражнения

1. Пусть задан двумерный массив A (4,4). Определить перечень и порядок следования элементов этого массива в следующих списках вводимых величин:

1) A

2) ((A(I, J), J = 1,4),I = 1,4)

3) ((A(K, L),K = 1,3),L = 2,4)

2. Пусть в фортран-программе имеется следующий оператор:

INTEGER A,B(10),C(2,7)

Написать список выводимых величин, не содержащий циклических элементов и эквивалентный следующему списку:

A,(B(K),K = 2,10,2),((C(I, J),I = 1,2), J = 1,2)

3. Переписать предлагаемый ниже список вводимых величин, используя циклический элемент:

B(2),C(2),B(5),C(5),B(8),C(8)

Глава 6

ОПЕРАТОРЫ БЕСФОРМАТНОГО ВВОДА-ВЫВОДА

Операторы бесформатного ввода-вывода, описанные в настоящей главе, предназначены для обмена данными между оперативной памятью и наборами данных прямого и последовательного доступов без преобразования форм их представления. В процессе описания операторов бесформатного ввода-вывода используются следующие обозначения: *n* — метка оператора ввода-вывода, *a* — ссылочный номер набора данных, *r* — порядковый номер записи данных, *c* — список вводимых или выводимых величин. Если не будет специальных оговорок, то под ссылочным номером набора данных следует понимать целую константу без знака или простую переменную целого типа стандартной длины, а под порядковым номером физической записи — целую константу без знака, переменную или арифметическое выражение целого типа.

6.1. Операторы бесформатного ввода-вывода прямого доступа

Операторы бесформатного ввода-вывода прямого доступа предназначены для обмена данными между наборами с прямым доступом и оперативной памятью. В фортран-программах могут использоваться наборы данных с прямым доступом, имеющие только формат F. К этой группе относятся следующие операторы: *оператор описания наборов данных*, *оператор бесформатного вывода прямого доступа*, *оператор бесформатного ввода прямого доступа* и *оператор поиска записи*.

Оператор описания наборов данных относится к невыполняемым операторам. С помощью этого оператора задаются некоторые характеристики наборов, которые необходимы для работы выполняемых операторов ввода-вывода. К таким характеристикам относятся количество физических записей в наборах и размеры физиче-

ских записей. Остальные операторы этой группы относятся к выполняемым операторам.

Наборы данных с прямой организацией обычно создаются заранее с помощью специальных программ. Если же это не сделано, то они создаются в процессе выполнения данной программы. Сведения о том, нужно ли создавать тот или иной набор данных в процессе выполнения программы и на каком устройстве его создавать, передаются через управляющие операторы задания. Наборы прямого доступа открываются при первом обращении к ним с помощью оператора ввода или оператора вывода, причем они открываются, как обновляемые. Закрытие наборов прямого доступа осуществляется без специальных указаний после окончания выполнения программы.

Вопросы и упражнения

1. Перечислить операторы, относящиеся к группе операторов бесформатного ввода-вывода прямого доступа.

2. Какой формат блоков должен иметь набор данных прямого доступа, если он используется в фортран-программе?

6.1.1. Оператор описания наборов для машинных форм представления данных. Этот оператор имеет вид:

DEFINE FILE $a_1(m_1, l_1, r_1, t_1)$ $a_2(m_2, l_2, r_2, t_2)$, . . ., $a_n(m_n, l_n, r_n, t_n)$

где a_i ($i = 1, 2, \ldots, n$) — целая константа без знака, означающая ссылочный номер набора данных; m_i — целая константа без знака, означающая количество физических записей (блоков) в наборе a_i; l_i — целая константа без знака, означающая длину физических записей (блоков) в наборе a_i. Единица измерения длины записей (блоков) и форма представления данных определяется параметром r_i. Для машинных форм представления данных параметр r_i может быть использован в одной из двух форм: L или U. Буква L означает, что длины измеряются в байтах (в позициях), и к набору данных можно обращаться как через операторы бесформатного ввода-вывода, так и через операторы форматного ввода-вывода. Буква U означает, что длины измеряются в словах (одно слово содержит 4 байта), и к набору данных можно обращаться только через операторы бесформатного ввода-вывода. Параметр t_i представляет собой простую переменную целого типа, которая называется *переменной связи*. После выполнения оператора ввода или вывода эта переменная принимает значение, равное порядковому номеру физической

записи, которая непосредственно следует за последней физической записью, участвовавшей в операции (порядковые номера записей исчисляются с 1). После выполнения оператора поиска записи эта переменная принимает значение, равное порядковому номеру найденной записи.

Например, оператор

```
DEFINE FILE 2(30,70,L,N1),3(100,15,U,N2)
```

описывает два набора данных со ссылочными номерами 2 и 3. Набор 2 состоит из 30 физических записей по 70 байтов (позиций) в каждой. В качестве переменной связи для этого набора используется переменная N1. Набор 3 состоит из 100 физических записей по 15 слов (60 байтов) в каждой. В качестве переменной связи для этого набора используется переменная N2.

Каждый из наборов прямого доступа должен быть описан только один раз. Такое описание должно располагаться в программе раньше других операторов ввода-вывода, относящихся к этому набору. Программа может содержать несколько операторов описания наборов данных.

Если программа состоит из нескольких программных модулей и имеет оверлейную структуру (с запланированным перекрытием), то все программные модули, содержащие описания наборов данных, должны включаться в корневой сегмент программы.

Переменную связи удобно использовать в операторах ввода и вывода для указания порядкового номера физической записи.

При разработке программ с модульной структурой необходимо учитывать то обстоятельство, что переменная связи локализована в том программном модуле, которому принадлежит оператор описания соответствующего набора данных. Если значение переменной связи используется в операторах обмена, принадлежащих другому программному модулю, то нужно позаботиться о передаче значения переменной связи в этот модуль. Эту операцию следует осуществлять через общую область, так как аппарат параметров для этой цели не всегда можно использовать.

Вопросы и упражнения

1. Написать операторы описания следующих наборов прямого доступа для машинных форм представления данных:

а) набора со ссылочным номером 4, содержащего 50 записей по 300 слов;

б) набора со ссылочным номером 5, состоящего из 1200 записей по 1210 байтов.

2. Указать, какие из приведенных ниже операторов описания наборов данных прямого доступа являются допустимыми:

а) DEFINE FILE K(11,25,U,I)

б) DEFINE FILE 11(40,250,U,MAK)

в) DEFINE FILE 1(60, 30,U,LAMA),2(200,300,L,K2),3(150,40, U,NIK)

г) DEFINE FILE 15(A,1024,L,KAMA),17(25,2000,A,J)

д) DEFINE FILE 27(1500,L,U,LY)

6.1.2. Оператор бесформатного вывода прямого доступа. Этот оператор имеет вид:

n WRITE (*a'r*)*c*

где *n* — либо пусто, либо метка, помечающая данный оператор; *a* — ссылочный номер набора данных; *r* — порядковый номер физической записи; *c* — список выводимых величин, совокупность значений которых образует логическую запись.

Оператор вывода осуществляет засылку в физическую запись с номером *r* значений величин, перечисленных в списке *c*. После заполнения физической записи *r* данные будут засылаться в физическую запись *r* + 1 и т. д. Этот процесс будет продолжаться до исчерпания списка *c* (до исчерпания логической записи).

Примеры операторов бесформатного вывода прямого доступа:

```
5 WRITE  (7'K)A, B, X
  WRITE (N'M + 1)X, Y(5),Z(K)
```

Рассмотрим различные случаи, возникающие в процессе выполнения оператора бесформатного вывода.

С л у ч а й 1. Длина логической записи *c* равна длине физической записи *r*. Все позиции физической записи *r* будут заполнены значениями величин, образующих логическую запись.

С л у ч а й 2. Длина логической записи *c* меньше длины физической записи *r*. Значения величин, образующих логическую запись *c*, будут размещены в начале физической записи *r*, а ее остаток будет заполнен восьмиразрядными двоичными кодами, состоящими из нулей.

С л у ч а й 3. Длина логической записи *c* больше длины физической записи *r*. Значения величин, образующих логическую запись *c*, будут размещены в нескольких подряд расположенных физических записях, начиная с той записи, номер которой указан в операторе вывода (*r*). При этом в каждой физической записи будет размещено максимально возможное количество значений величин. Если значение некоторой величины не помещается в текущую фи-

ическую запись, то оно полностью переносится в следующую физическую запись, а остаток текущей физической записи заполняется восьмиразрядными двоичными кодами, состоящими из нулей. Остаток последней физической записи также будет заполнен восьмиразрядными двоичными кодами, состоящими из нулей. Таким образом, в результате выполнения оператора бесформатного вывода одна логическая запись займет в наборе данных целое количество физических записей (одну или несколько). При этом переменная связи, указанная в операторе описания набора данных, примет значение, равное $r + i$, где i — количество физических записей, занятых логической записью c. Заметим, что физические записи, содержащие сегменты логических записей, ничем не отличаются от физических записей, содержащих полные логические записи.

Рассмотрим несколько примеров.

Пример 1. Пусть задана программа:

```
REAL *8X(40)/40*5.0/,Y(20)/20*7.0/
INTEGER  K
DEFINE FILE 9(4,480,L,K)
WRITE (9'2)X,Y
RETURN
END
```

В результате выполнения этой программы во вторую физическую запись 9-го набора будут помещены значения переменных X (1), X (2), . . ., X (40), Y (1), Y (2), . . ., Y (20), которые займут в ней все 480 байтов.

После выполнения оператора WRITE переменная связи K будет иметь значение, равное трем.

Пример 2. Пусть задана программа:

```
REAL *8X(40)/40*5.0/,Y(20)/20*7.0/
INTEGER K
DEFINE FILE 9(40,500,L,K)
K = 2
WRITE (9'K + 1)X,Y
RETURN
END
```

В результате выполнения этой программы в третью физическую запись набора 9 будут помещены значения величин X(1), X(2), ... X(40), Y(1), Y(2), . . ., Y(20), которые займут в ней 480 байтов, оставшиеся 20 байтов будут заполнены нулевыми кодами.

После выполнения оператора WRITE переменная связи будет иметь значение, равное четырем.

Пример 3. Пусть задана программа:

```
REAL *8X(20)/20*1.0/,Y(20)/20*2.0/
INTEGER K,M,N
DEFINE FILE 9(4,130,L,K)
M = 1
WRITE (9'M)(X(N),Y(N),N=1,20)
RETURN
END
```

В результате выполнения этой программы в первую физическую запись 9-го набора будут помещены значения величин X(1), Y(1), X(2), Y(2), . . ., X(8), Y(8), которые займут в ней 128 байтов. Оставшиеся 2 байта будут заполнены нулевыми кодами. Во вторую физическую запись 9-го набора будут помещены значения величин X(9), Y(9), X(10), Y(10), . . ., X(16), Y(16), которые займут в ней 128 байтов. Оставшиеся два байта будут заполнены нулевыми кодами. В третью физическую запись 9-го набора будут помещены значения величин X(17), Y(17), X(18), Y(18), X(19), Y(19), X(20), Y(20), которые займут в ней 64 байта. Оставшиеся 66 байтов будут заполнены нулевыми кодами. После выполнения оператора WRITE переменная связи K будет иметь значение, равное четырем.

Пример 4. Пусть задана программа:

```
      REAL *8X(20)/20*1.0/,Y(20)/20*2.0/
      INTEGER M,N
      DEFINE FILE 5(4,130,L,M)
      M = 1
1     WRITE (5'M)(X(N),Y(N),N=1,16)
2     WRITE (5'M)(X(N),Y(N),N=17,20)
      RETURN
      END
```

Первый оператор вывода этой программы (1 WRITE...) поместит значения переменных X(1), Y(1), X(2), Y(2), . . ., X(8), Y(8) в первую физическую запись 5-го набора, а значения переменных X(9), Y(9), X(10), Y(10), . . ., X(16), Y(16) — во вторую физическую запись 5-го набора. После этого переменная связи M примет значение, равное 3. Второй оператор вывода (2 WRITE...) поместит значения переменных X(17), Y(17), X(18), Y(18), X(19), Y(19), X(20), Y(20) в третью физическую запись 5-го набора, после чего переменная связи M примет значение, равное 4. В оставшиеся позиции пер-

вой, второй и третьей физических записей будут помещены восьмираарядные двоичные коды, состоящие из нулей.

Пример 5. Пусть задана программа модульной структуры, состоящая из основной программы и подпрограммы P:

```
      REAL *8X(20)/20*1.0/,Y(20)/20*2.0/
      INTEGER M
      DEFINE FILE 7(4, 130, L, M)
      M = 1
      CALL P(X,Y,M)
      RETURN
      END
      SUBROUTINE P(X1,Y1,M1)
      REAL *8X1(20),Y1(20)
      INTEGER M1, N1
    1 WRITE (7'M1)(X1(N1), Y1(N1), N1 = 1,16)
    2 WRITE (7'M1) (X1(N1),Y1(N1), N1=17,20)
      RETURN
      END
```

Первый оператор вывода (1 WRITE...), входящий в состав подпрограммы P, поместит значения переменных X1(1), Y1(1), X1(2), Y1(2), . . ., X1(8), Y1(8) в первую физическую запись 7-го набора, а в оставшиеся позиции этой записи поместит восьмираазрядные двоичные коды, состоящие из нулей. Этот же оператор поместит значения переменных X1(9), Y1(9), X1(10), Y1(10), . . ., X1(16), Y1(16) во вторую физическую запись 7-го набора, а в оставшиеся позиции этой записи поместит восьмираазрядные двоичные коды, состоящие из нулей. Переменная связи M, локализованная в основной программе, примет значение, равное 3. Значение переменной M1, локализованной в подпрограмме P, останется равным 1.

Второй оператор вывода, входящий в состав подпрограммы P (2 WRITE...), поместит значения переменных X1(17), Y1(17), X1 (18), Y1 (18), X1 (19), Y1 (19), X1 (20), Y1 (20) в первую запись 7-го набора, а в оставшиеся позиции этой записи поместит восьмиразрядные двоичные коды, состоящие из нулей. После этого переменная связи M, локализованная в основной программе, примет значение, равное 2.

Вопросы и упражнения

1. Пусть заданы набор данных прямого доступа со ссылочным номером 2 и одномерные массивы XM и XY, относящиеся к вещественному типу стандартной длины. Набор данных состоит из трех

физических записей длиной по 100 слов. Массив XM содержит 100 элементов, каждый из которых равен числу 1,75. Массив XY содержит 200 элементов, каждый из которых равен числу 2,75. Составить программу засылки значений элементов массивов XM и XY в набор 2 таким образом, чтобы значения элементов массива XM попали в первую его запись, а значения элементов массива XY — во вторую и третью записи.

2. Пусть задан двумерный массив S1, относящийся к целому типу стандартной длины. Этот массив содержит 8 строк и 10 столбцов, причем значения всех его элементов равны числу 5. Определить минимальные значения характеристик набора данных прямого доступа (количество и длину физических записей) со ссылочным номером 10 и составить программу засылки значений элементов массива S1 в набор 10 таким образом, чтобы каждая строка массива поместилась в отдельную физическую запись набора. Порядковый номер физической записи должен быть равен порядковому номеру помещенной в нее строки массива.

3. Пусть заданы одномерный массив MAS, содержащий 50 элементов, переменные A, B, C, D и набор данных прямого доступа со ссылочным номером 3, содержащий 11 физических записей длиной по 15 слов. Массив и переменные относятся к вещественному типу нестандартной длины. Составить фрагмент программы для засылки в набор 3 значений перечисленных выше величин таким образом, чтобы в первую запись набора попали значения первых пяти элементов массива MAS; во вторую запись — значения переменных A, B, C, D; в третью и последующие записи — значения остальных элементов массива MAS по пять значений в каждую запись.

4. Пусть заданы три одномерных массива A, B и C, каждый из которых содержит по 54 элемента и относится к вещественному типу стандартной длины, а также набор прямого доступа со ссылочным номером 10, состоящий из семи физических записей длиной по 60 слов.

Составить фрагмент программы для засылки значений элементов этих массивов в набор 10 таким образом, чтобы в третью запись набора попали значения элементов A(1), B(1), C(1), A(2), B(2), C(2), . . ., A(18), B(18), C(18); в четвертую — значения элементов A(19), B(19), C(19), A(20), B(20), C(20), . . ., A(36), B(36), C(36) и т. д. Значения элементов массивов должны размещаться в записях в указанном выше порядке.

5. Пусть заданы набор прямого доступа со ссылочным номером 9, содержащий 50 физических записей длиной по 51 байту, а

также одномерный массив MAC1, относящийся к целому типу нестандартной длины и содержащий 250 элементов.

Составить фрагмент программы, обеспечивающий:

а) вычисление новых значений элементов массива MAC1, которые должны быть равны их старым значениям, помноженным на порядковые номера этих элементов;

б) засылку новых значений элементов массива MAC1 в набор 9 таким образом, чтобы значения первых 25 его элементов попали в десятую запись набора, значения следующих 25 элементов — в одиннадцатую запись набора и т. д.

6.1.3. Оператор бесформатного ввода прямого доступа. Этот оператор имеет вид:

n READ (*a'r*)*c*

где *n* — либо пусто, либо метка, помечающая данный оператор; *a* — ссылочный номер набора данных; *r* — порядковый номер физической записи (блока); *c* — список вводимых величин.

Оператор ввода осуществляет выборку значений величин из физической записи с порядковым номером *r* и присваивание этих значений величинам, перечисленным в списке *c*. Очередность присваивания значений величинам, содержащимся в списке *c*, определяется порядком их следования в списке *c*.

После исчерпания физической записи *r* данные будут выбираться из физической записи *r* + 1 и т. д. Этот процесс будет продолжаться до тех пор, пока не будут присвоены значения всем величинам из списка *c*. Если в процессе выборки значений будет создана такая ситуация, что количество оставшихся в записи байтов окажется меньше длины очередной величины (величины из списка *c*, которой нужно присвоить значение), то эти байты будут пропущены, а в качестве значения очередной величины будут браться данные из следующей записи.

Присваивание значений величинам из списка *c* означает их размещение в тех областях (полях) оперативной памяти, которые предназначены для хранения значений этих величин.

Рассмотрим несколько примеров.

Пример 1. Пусть задана программа:

```
      INTEGER A(8)/11,12,13,14,15,16,17,18/,B(9),M,N,L
      DEFINE FILE 7(4,18,L,N)
      WRITE (7'2)A
      N = 2
    1 READ (7'N)B(1),B(2)
```

```
    2 READ (7'N)B(3),B(4)
    3 READ (7'2)  (B(N),N=5,9)
      RETURN
      END
```

После выполнения оператора WRITE (7'2)A вторая физическая запись 7-го набора будет содержать числа 11, 12, 13 и 14. Каждое из этих чисел будет представлено в машинной форме и будет занимать в записи по 4 байта. Последние 2 байта будут содержать восьмиразрядные двоичные коды, состоящие из нулей. Третья физическая запись будет содержать числа 15, 16, 17 и 18. Каждое из этих чисел также будет представлено в машинной форме и будет занимать в записи по 4 байта. Последние 2 байта этой записи будут содержать восьмиразрядные двоичные коды, состоящие из нулей.

После выполнения операторов 1 READ..., 2 READ... и 3 READ... элементы массива B будут иметь следующие значения: B(1) = 11, B(2) = 12, B(3) = 15, B(4) = 16, B(5) = 11, B(6) = = 12, B(7) = 13, B(8) = 14, B(9) = 15.

Пример 2. Пусть задана программа модульной структуры, состоящая из основной программы и двух подпрограмм SUBW и SUBR:

```
      REAL N1(50)/10*0.5,10*0.1,10*0.3,10*0.7,10*0.6/,N2(20
           /10*0.01,10*0.15/,M1(5,10),M2(2,10)
      DIMENSION TM1 (10, 5), TM2(10,2)
      COMMON (L)
      DEFINE FILE 2(15,10,U,L)
      L = 1
      CALL SUBW(N1,50)
      CALL SUBW(N2,20)
      L = 1
      CALL SUBR(M1,5,10)
      DO 1 J=1,5
      DO 1 K=1,10
    1 TM1(K, J)=M1(J,K)
      CALL SUBR(M2,2,10)
      DO 2 J=1,2
      DO 2 K=1,10
    2 TM2(K,J)=M2(J,K)
    3 WRITE (2'L)((TM1(I,J),J=1,5),I=1,10)
    4 WRITE (2'L)((TM2(I,J),J=1,2),I=1,10)
      RETURN
      END
```

```
SUBROUTINE SUBW(B,N)
COMMON (L2)
DIMENSION B(N)
WRITE (2'L2)B
RETURN
END
SUBROUTINE SUBR (C,N,M)
COMMON (L1)
DIMENSION C(N,M)
READ (2'L1) ((C(I, J), J = 1, M), I = 1, N)
RETURN
END
```

Подпрограмма SUBW осуществляет запись во 2-й набор одномерного массива, имя которого и размер передаются из основной программы через параметры. При этом номер физической записи, с которой начинается размещение данных в набор, определяется значением переменной L2. Количество физических записей, которые будут отведены под величины, образующие логическую запись, определяется размером массива B.

Подпрограмма SUBR осуществляет чтение из 2-го набора значений элементов в двумерный массив, имя и размеры которого передаются из основной программы через параметры. Номер физической записи, с которой начинается выборка данных, определяется значением переменной L1. Количество записей, на которых осуществляется чтение данных, зависит от размеров массива, значения элементов которого читаются из 2-го набора.

Заметим, что в основной программе переменная связи 2-го набора L размещена в общей области. Переменные L2 в подпрограмме SUBW и L1 в подпрограмме SUBR размещены в той же общей области и в том же поле, т. е. принимают то же значение, что переменная L.

Другими словами, значение переменной связи L передается подпрограммам SUBW и SUBR через общую область.

При выполнении основной программы в результате 1-го обращения к подпрограмме SUBW в набор данных 2 будут записаны значения элементов массива N1, которые будут размещены в первых пяти физических записях. Переменная связи L примет значение, равное 6. После 2-го обращения к подпрограмме SUBW во 2-й набор данных будут помещены значения массива N2 в 6-ю и 7-ю записи. Переменная связи L будет равна 8. Перед 1-м обращением к подпрограмме SUBR переменной L в основной програм-

ме присваивается значение, равное 1. Поэтому после выполнения оператора CALL SUBR (M1, 5, 10) в массив M1 будут считаны данные на первых пяти записей. В результате элементы массива M1 будут иметь следующие значения:

M1(1,1)=M1(1,2)=...=M1(1,10)=0.5,
M1(2,1)=M1(2,2)=...=M1(2,10)=0.1,
M1(3,1)=M1(3,2)=...=M1(3,10)=0.3,
M1(4,1)=M1(4,2)=...=M1(4,10)=0.7,
M1(5,1)=M1(5,2)=...=M1(5,10)=0.6.

После выполнения оператора READ переменная связи примет значение, равное 6. В результате 2-го обращения к подпрограмме SUBR переменная связи L примет значение, равное 8, а элементы массива M2 будут иметь следующие значения:

M2(1,1)=M2(1,2)=...=M2(1,10)=0.01,
M2(2,1)=M2(2,2)=...=M2(2,10)=0.15.

После выполнения оператора 3 WRITE значения элементов массива TM1 будут помещены в физические записи 8,9,10,11,12,а переменная связи примет значение 13. Оператор 4 WRITE поместит значения элементов массива TM2 в физические записи 13 и 14.

Вопросы и упражнения

1. Пусть задан набор данных прямого доступа с номером 1, состоящий из 720 записей. Первая запись содержит числа:

—3.21, 0.1, 6.15, —15.0, —0.2, 15.1, 4.2;

вторая запись содержит числа:

21.01, 6.13, —14,2, —1.3, 6.4.

Эти числа имеют стандартную длину, относятся к вещественному типу и представлены в машинной форме.

Определить значения переменных A, B, Q, R, S, которые они получат в результате выполнения следующей программы:

```
REAL A,B,Q,R,S
DEFINE FILE 1(720,370,L,J)
READ (1'2)A,Q
READ (1'1)B,R,S
STOP
END
```

2. Пусть задан набор данных прямого доступа с номером 6, состоящий из 10 записей. Первая запись содержит числа:

—1096.3, 2.1, 7.2, —8.0;

вторая — числа:

9.6, 12.0, 18.1,—21.01, 0.109, 0.16, —22.9, —13.11, 0.999, 0.100, 1.101, 202.0.

Эти числа имеют стандартную длину, относятся к вещественному типу и представлены в машинной форме.

Определить значения элементов массива A, которые они принимают в результате выполнения следующей программы:

```
REAL *4A(10)/—64.1,11.2,3*0.3,6.4,—17.21,2*8.1,—17.01/
DEFINE FILE 6(10,400,L,L)
READ (6'2)(A(J),J=3,8,2)
STOP
END
```

3. Пусть 7-я запись набора данных с номером 5 содержит числа:

—709.8,6.18, 29.302, 11.16, —9.34,8.01, 109.3, 394.21, —6.28, 29.01, —35.0, 43.09, 69.01, —79.6, —59.4, 0.1015, 111.2, 302.6, —400. 1,0.12, 11.3, —16.2, —4.2, 4.01.

Эти числа имеют стандартную длину, относятся к вещественному типу и представлены в машинной форме.

Определить значения элементов массива X, которые они получат в результате выполнения следующей программы:

```
REAL   X(4,5)/—0.17,1.02,3*—6.18,  4*0.28,  —19.04,3.08,3*
       —0.501E03,2.18E—1,5*6.13/
DEFINE FILE 5(20,800,L,J)
READ (5'7)((X(K,L),L=3,5,2), K=2,4)
STOP
END
```

4. Пусть заданы:

а) набор данных прямого доступа со ссылочным номером 1, состоящий из пяти физических записей длиной по 10 слов;

б) одномерный массив MD, состоящий из десяти элементов MD(1) = 1, MD(2) = 2, . . ., MD(10) = 10;

в) переменные K, N и одномерный массив MR, состоящий из восьми элементов.

Все величины (массивы и переменные) относятся к целому типу стандартной длины.

Составить программу, обеспечивающую:

а) засылку элементов массива MD в первую запись набора;

б) чтение значений элементов массива MD из первой записи набора и присваивание этих значений переменным K, N и элементам массива MR.

Переменные K, N и элементы массива MR должны получить следующие значения: N = MD(9), K = MD(10), MR(1) = MD(1), MR(2) = MD(2), ..., MR(8) = MD(8).

5. Пусть заданы:

а) набор данных прямого доступа со ссылочным номером 2, состоящий из 3-х физических записей длиной по шесть слов;

б) одномерный массив MW, состоящий из 15-ти элементов MW(1) = 11, MW(2) = 12, ..., MW(5) = 15, MW(6) = 21, MW(7) = = 22, ..., MW(10) = 25, MW(11) = 31, MW(12) = 32, ..., MW(15) = 35;

в) одномерные массивы M1, M2 и M3, состоящие из семи, пяти и трех элементов соответственно. Все массивы относятся к целому типу стандартной длины.

Составить программу, обеспечивающую:

а) засылку в набор данных значений элементов массива MW таким образом, чтобы в первую запись набора попали значения элементов MW(1), MW(2), ..., MW(5); во вторую запись — значения элементов MW(6), MW(7), ..., MW(10); в третью запись — значения элементов MW(11), MW(12), ..., MW(15);

б) чтение значений элементов массива MW из набора 2 и присваивание этих значений элементам массивов M1, M2 и M3.

Элементы этих массивов должны получить следующие значения:

M1(1) = MW(1), M1(2) = MW(2), ..., M1(7) = MW(7);

M2(1) = MW(8), M2(2) = MW(9), ..., M2(5) = MW(12);

M3(1) = MW(13), M3(2) = MW(14), M3(3) = MW(15).

6. Пусть заданы:

а) набор данных прямого доступа со ссылочным номером 9, содержащий 30 физических записей длиной по 10 слов; 15 записей этого набора, начиная с 16-й, содержат по 10 значений переменных вещественного типа стандартной длины;

б) двумерный массив MATR вещественного типа стандартной длины, состоящий из 15 строк и 10 столбцов.

Составить программу, обеспечивающую чтение значений переменных из набора 9 и присваивание этих значений элементам мас-

сива MATR. Элементам первой строки массива должны быть присвоены значения переменных, содержащиеся в 16-й записи, элементам второй строки — значения переменных, содержащиеся в 17-й записи и т. д.

7. Пусть заданы:

а) набор данных прямого доступа со ссылочным номером 15, состоящий из 22-х физических записей длиной по 20 слов;

б) матрица (двумерный массив) B вещественного типа стандартной длины, состоящая из 20 строк и 20 столбцов;

в) векторы (одномерные массивы) A и C вещественного типа стандартной длины, каждый из которых состоит из 20 элементов.

Значения элементов вектора A содержатся в 1-й записи набора 15, а значения элементов матрицы B — во второй записи (1-я строка), в третьей записи (2-я строка) и т. д.

Составить программу, обеспечивающую вычисление значений элементов вектора C по формуле

$$C_i = \sum_{j=1}^{20} B_{ij} A_j \quad (i = 1, 2, \ldots, 20)$$

и засылку этих значений в 22-ю запись набора 15.

8. Пусть заданы два набора данных прямого доступа со ссылочными номерами 10 и 11. Каждый из этих наборов содержит по 15 записей длиной по 101-му байту. Составить программу переписи содержимого 10-го набора в 11-й набор. Порядок размещения данных в наборе 11 должен быть таким же, что и в наборе 10.

6.1.4. Оператор поиска записи. Оператор поиска записи имеет вид:

n FIND (*a*'*r*)

где *n* — либо пусто, либо метка, помечающая данный оператор; *a* — ссылочный номер набора данных; *r* — порядковый номер физической записи в наборе *a*.

Под воздействием этого оператора устройство ввода-вывода, на котором находится носитель информации с набором данных *a*, устанавливается в такое положение, при котором требуется минимальное время на последующее обращение к записи *r*. Под обращением к записи понимается выборка данных из записи или засылка данных в запись.

Поиск записи осуществляется одновременно с выполнением операторов, следующих за оператором поиска и не имеющих обра-

щений к устройству ввода-вывода, на котором размещается набор данных *a*. Оператор поиска записи прекращает выполняться в следующих случаях:

1) когда запись будет найдена (когда устройство ввода-вывода будет установлено в соответствующее положение);

2) когда начнет выполняться другой оператор ввода-вывода, имеющий обращение к устройству, на котором размещается набор данных *a*.

Под обращением к устройству понимается выборка данных или засылка данных в любой набор, расположенный на данном устройстве.

Оператор поиска записи используется для ускорения процесса выполнения программы, так как на поиск нужной записи расходуется значительное время (по сравнению с выборкой или записью данных). Ускорение процесса вычислений осуществляется только за счет одновременного выполнения оператора поиска записи и других операторов, следующих (по времени их исполнения) за оператором поиска.

Рассмотрим пример. Пусть набор данных со ссылочным номером 15 состоит из 25 записей, каждая из которых содержит по 400 чисел. Эти числа относятся к вещественному типу и имеют стандартную длину (т. е. занимают в записи по 4 байта каждое). Необходимо вычислить функцию $c = \sum_{i=1}^{10000} \sin(x_i)$, где $x_1, x_2, \ldots, x_{10000}$ — совокупность чисел, содержащихся в наборе данных с номером 15.

Эту функцию можно вычислить по следующей программе:

```
      REAL X1(400),C
      INTEGER K,M,N
      DEFINE FILE 15(25,400,U,K)
      C=0
      DO 1 M=1,25
      READ (15'M)X1
      DO 1 N=1,400
    1 C=C+SIN(X1(N))
      RETURN
      END
```

Программа будет исполняться быстрее, если в нее после оператора READ (15'M)X1 добавится оператор поиска записи FIND (15'M + 1)

6.2. Операторы бесформатного ввода-вывода последовательного доступа

К этой группе относятся операторы, предназначенные для обмена данными между оперативной памятью и наборами данных с последовательной организацией без преобразования форм представления данных. При этом могут использоваться только такие наборы, блоки которых имеют переменную длину и допускают сегментирование логических записей (форматы VS и VBS).

К этой группе операторов относятся: *оператор бесформатного вывода последовательного доступа*, *оператор бесформатного ввода последовательного доступа*, *оператор возврата на одну запись*, *оператор возврата на первую запись*. Все эти операторы относятся к категории выполняемых.

Набор с последовательной организацией открывается при первом обращении к нему с помощью операторов ввода или вывода. Он открывается как входной, если первое обращение к нему осуществляется с помощью оператора ввода, и как выходной, если первое обращение к нему осуществлялось с помощью оператора вывода. Закрытие набора с последовательной организацией осуществляется либо в процессе выполнения оператора возврата на первую запись, либо после окончания выполнения программы.

Форматы блоков, максимальные длины блоков, максимальные длины физических записей и другие характеристики наборов данных, необходимые для выполнения операторов ввода-вывода, указываются в управляющих операторах задания.

Вопросы и упражнения

1. Назвать форматы блоков для наборов последовательного доступа, к которым можно обращаться через операторы бесформатного ввода-вывода.

2. Перечислить операторы, входящие в группу операторов бесформатного ввода-вывода последовательного доступа.

6.2.1. Оператор бесформатного вывода последовательного доступа. Этот оператор может представляться в одной из следующих форм:

n WRITE (*a*) *c*
n WRITE (*a*)

где *n* — либо пусто, либо метка, помечающая данный оператор; *a* — ссылочный номер набора данных; *c* — список выводимых величин, совокупность значений которых образует логическую запись.

Здесь и во всех остальных операторах бесформатного ввода-вывода последовательного доступа ссылочный номер набора данных может указываться в виде целого числа без знака и в виде простой переменной целого типа стандартной длины. Так, например, оператор бесформатного вывода последовательного доступа может иметь вид:

```
37 WRITE (9) A,B,C
```

или

```
WRITE(N)Y(5),(Z(K),K=1,10)
```

Оператор бесформатного вывода последовательного доступа выполняет следующие функции:

1) Открывает набор данных *a* как выходной. Процедура открытия выполняется только при первом обращении к набору после начала работы программы или же при первом обращении к набору после его закрытия.

2) Создает в наборе *a* очередную логическую запись и засылает в нее значения величин, перечисленных в списке *c*. При первом обращении к набору создается первая логическая запись, которая размещается в начале набора, при втором обращении — вторая логическая запись, которая размещается следом за первой и т. д. Таким образом, создание набора данных с последовательной организацией осуществляется в процессе записи данных в этот набор. Если список выводимых величин отсутствует, то создается фиктивная логическая запись. При чтении данных фиктивные записи должны пропускаться.

Если набор данных был открыт оператором бесформатного вывода, то в дальнейшем (до его закрытия) данные в этот набор должны засылаться только оператором бесформатного вывода. Рассмотрим различные случаи, возникающие при создании наборов данных, имеющих форматы VS и VBS.

Случай 1. Блоки набора данных имеют формат VS, длина логической записи (в байтах) меньше или равна $l - 8$ (l — максимальное количество байтов в блоке). В этом случае логическая запись будет оформлена в виде одного сегмента, который будет помещен в отдельный (очередной) блок. Длина сегмента будет равна $l_1 + 4$ (l_1 — длина логической записи).

Случай 2. Блоки набора данных имеют формат VS, длина логической записи (в байтах) больше, чем $l - 8$ (l — максимальное количество байтов в блоке). В этом случае логическая запись будет разделена на сегменты, каждый из которых будет помещен в отдельный блок. Деление на сегменты осуществляется таким обра-

зом, чтобы каждый сегмент (за исключением последнего) включал в себя как можно больше значений величин из логической записи, и его длина l_c удовлетворяла условию $l_c \leqslant l - 4$.

Случай 3. Блоки набора данных имеют формат VBS. В этом случае разбиение логической записи на сегменты и размещение этих сегментов в блоках набора данных осуществляется по таким же правилам, что и в случаях 1 и 2. Особенность заключается в следующем. Если длина последнего заполненного блока (при выполнении предыдущего оператора вывода) оказалась меньше максимально возможной, то первый сегмент данной логической записи будет помещен в этот же блок следом за последним сегментом предыдущей логической записи, и его длина будет скорректирована. Это обстоятельство учитывается при разбиении логической записи на сегменты.

Рассмотрим несколько примеров.

Пример 1. Пусть имеется программа:

```
REAL A(50)/50 * 6.0/,B(40) * 8/40 * 0.7D1/
WRITE (9) A
WRITE (9) B
RETURN
END
```

Программа создает набор данных с последовательной организацией. Ссылочный номер набора данных — 9, формат блоков — VS, максимальная длина блока — 144 байта. В результате выполнения этой программы в наборе данных с номером 9 будут созданы две логические записи. Первая из них будет содержать 50 значений элементов массива A, вторая — 40 значений элементов массива B.

Первая логическая запись будет состоять из двух сегментов и будет размещена в двух блоках (первом и втором), вторая будет состоять из трех сегментов и будет размещена в трех блоках (третьем, четвертом и пятом). Характеристики созданного набора данных приведены в табл. 6.1.

Пример 2. Пусть задана программа:

```
REAL A(50)/50 * 6.0/,B(35)/35 * 0.7/,C(5)/5 * 3,0/
WRITE(9)A
WRITE(9)B
WRITE(9)C
RETURN
END
```

Таблица 6.1. Характеристики набора данных для примера 1

Номер логической записи	Номер блока	Длина блока в байтах	Длина сегмента в байтах	Длина поля данных в байтах	Содержимое поля данных
1	1	144	140	136	с A(1) по A(34)
	2	72	68	64	с (A35) по A(50)
2	3	144	140	136	с B(1) по B(17)
	4	144	140	136	с B(18) по B(34)
	5	56	52	48	с B(35) по B(40)

Таблица 6.2. Характеристики набора данных для примера 2

Номер блока	Длина блока в байтах	Номер сегмента	Длина сегмента в байтах	Длина поля данных в байтах	Содержимое поля данных	Номер логической записи
1	68	1	64	60	с A(1) по A(15)	1
2	68	2	64	60	с A(16) по A(30)	
3	68	3	64	60	с A(31) по A(45)	
4	68	4	24	20	с A(46) по A(50)	
		1	40	36	с B(1) по B(9)	2
5	68	2	64	60	с B 10) по B(24)	
6	68	3	48	44	с B(25) по B(35)	
		1	16	12	с C(1) по C(3)	3
7	16	2	12	8	с C(4) по C(5)	

Эта программа создает набор данных с последовательной организацией. Ссылочный номер набора данных — 9, формат блоков VBS, максимальная длина блока — 68 байтов. В результате выполнения этой программы в 9-м наборе данных будут созданы 3 логические записи, первая из которых будет содержать значения элементов массива A, вторая — значения элементов массива B,

а третья — значения элементов массива С. Первая логическая запись будет состоять из четырех сегментов. Первые три из них будут размещены в отдельных блоках с порядковыми номерами 1,2,3, а четвертый войдет в состав блока с порядковым номером 4. Вторая логическая запись будет состоять из трех сегментов. Первый сегмент войдет в состав четвертого блока. Второй сегмент будет размещен в пятом блоке, а третий войдет в состав шестого блока. Третья логическая запись будет разделена на два сегмента: первый войдет в состав шестого блока, второй будет размещен в седьмом блоке. Характеристики созданного набора данных приведены в табл. 6.2.

Вопросы и упражнения

1. Пусть заданы:

а) Набор данных прямого доступа со ссылочным номером 17, содержащий 2 физические записи длиной по 50 слов. В каждой из этих записей хранится по 50 целых чисел стандартной длины.

б) Набор данных последовательного доступа со ссылочным номером 16. Блоки этого набора имеют формат VS и максимальную длину, равную 188 байтам.

Составить программу переписи чисел из набора 17 в набор 16 таким образом, чтобы все числа вошли в состав одной (первой) логической записи набора 16. Определить количество и длину блоков набора данных 16, которые будут заняты этими числами.

2. Пусть заданы:

а) Три массива вещественных чисел нестандартной длины, из которых одномерный массив RM1 содержит 100 элементов, двумерный массив RM2 содержит 10 строк и 5 столбцов, двумерный массив RM3 — 7 строк и 10 столбцов.

б) Набор данных последовательного доступа со ссылочным номером 2. Блоки этого набора имеют формат VBS и максимальную длину, равную 612 байтам.

Составить фрагмент программы, обеспечивающий засылку значений элементов массива RM1 в первую логическую запись набора, а значений элементов массивов RM2 и RM3 — во вторую логическую запись набора. Определить количество и длину блоков набора 2, которые будут заняты значениями элементов этих массивов.

3. Пусть заданы:

а) Набор данных прямого доступа со ссылочным номером 9, содержащий 10 физических записей длиной по 60 слов. В каждой из этих записей хранится по 60 вещественных чисел стандартной длины.

б) Набор данных последовательного доступа со ссылочным номером 1. Блоки этого набора имеют формат VBS и максимальную длину, равную 50 байтам.

в) Одномерный массив MS1 вещественных чисел стандартной длины, содержащий 50 элементов.

Составить программу, обеспечивающую:

а) чтение 50-ти чисел из 5-й записи набора 9 и присваивание их в качестве значений элементам массива MS1;

б) пересчет значений первых десяти элементов массива MS1 по формуле

$$MS1_i = MS1_i^3 + 0.5$$

где через $MS1_i$ обозначен i-й элемент массива MS1;

в) засылку значений первых десяти элементов массива MS1 в первую логическую запись набора 1 и следующих сорока элементов массива — во вторую логическую запись набора 1.

Определить количество и длину блоков набора 1, которые будут заняты значениями элементов массива MS1.

6.2.2. Оператор бесформатного ввода последовательного доступа. Оператор бесформатного ввода последовательного доступа может представляться в одной из следующих форм:

```
n READ (a,END=m,ERR=k)c
n READ (a,END=m,ERR=k)
n READ (a,END=m)c
n READ (a,END=m)
n READ (a,ERR=k)c
n READ (a,ERR=k)
n READ (a)c
n READ (a)
```

Здесь n — либо пусто, либо метка, помечающая данный оператор; a — ссылочный номер набора данных; m, k — метки выполняемых операторов; c — список вводимых величин.

Примеры операторов ввода:

```
3 READ (4,END=5,ERR=8)(M(K),K=N,J)
  READ (I,END=3)X,Y,(Z(J),J=1,10),W
  READ (5)
```

Оператор бесформатного ввода выполняет следующие функции:

1) Открывает набор данных a как входной. Процедура открытия выполняется только при первом обращении к набору после на-

чала работы программы или же при первом обращении к набору после его закрытия.

2) Осуществляет выборку данных из очередной логической записи набора *a* и присваивает их величинам, перечисленным в списке вводимых величин. При первом обращении выборка будет осуществляться из первой логической записи, при втором — из второй и т. д. Объем порции данных, определяемой списком вводимых величин, не должен превышать объема логической записи. Если список *c* отсутствует, то очередная логическая запись будет пропущена.

3) Передает управление оператору, помеченному меткой *m*. Это происходит только при обнаружении конца набора данных (когда будет прочитан признак конца набора). В этом случае работу с набором можно продолжить только после его закрытия с помощью оператора возврата на первую запись. Если параметр END = *m* отсутствует, то при обнаружении признака конца набора данных выполнение программы будет прекращено.

4) Прерывает процедуру чтения и передает управление оператору с меткой *k*. Это происходит только при обнаружении ошибки в процессе чтения данных из набора в буфер. Если параметр ERR = *k* отсутствует, то при обнаружении ошибки выполнение программы будет прекращено.

В качестве примера рассмотрим следующую задачу. Пусть заданы:

а) Набор данных последовательного доступа, состоящий из двух логических записей. Ссылочный номер набора — 7; первая логическая запись содержит числа: 23.5, 2.99, —8.15, 17.99, 18.0, 0.08; вторая запись — числа: 0.001, — 375.18, 145.37, 845.7, — 444.01. Эти числа относятся к вещественному типу, имеют стандартную длину и представлены в машинной форме.

б) фортран-программа

```
REAL X,Y,Z,P,Q,L,M,N
READ (7)X,Y,Z
READ (7)Q,L
RETURN
END
```

Оператор чтения READ (7)X,Y,Z этой программы откроет набор данных со ссылочным номером 7 и выберет в буфер числа из первой логической записи. Первые три числа он присвоит в качестве значений переменным X,Y,Z, в результате чего они станут равными: X = 23.5, Y = 2.99, Z = —8.15. Оператор чтения READ (7)Q,L

выберет в буфер числа из второй логической записи. Первые два числа он присвоит в качестве значений переменным Q и L. Эти переменные станут равными: Q = 0.001, L = —375.18.

Вопросы и упражнения

1. Пусть имеется набор данных последовательного доступа с номером 10, состоящий из трех логических записей. Первая запись содержит числа:

22.7, 6.34, 15.2, —4.8, 6.8, —9.18, —34.09;

вторая запись содержит числа:

—38.01, 10.17, 9.64,—22.3, 5.18;

третья запись содержит числа:

107.07,—20.99, —9.601.

Эти числа имеют стандартную длину, относятся к вещественному типу и представлены в машинной форме. Определить значения элементов массива X, переменных Y, Z, которые они получат в результате выполнения следующей программы:

```
REAL Y,X(6),Z
READ (10)X
READ (10)Z
READ (10)Y
RETURN
END
```

2. Пусть имеется набор данных последовательного доступа с номером 14, состоящий из двух логических записей. Первая запись содержит числа:

20,0,0, —4,28;

вторая запись содержит числа:

—212, 16, —5, 1081, —716, 14, 15, —8, 16, 21.

Эти числа имеют нестандартную длину, относятся к целому типу и представлены в машинной форме.

Определить значения элементов массива X после выполнения следующей программы:

```
INTEGER *2X(10)/20,—3,3*6,—7,10,3,2*8/,K
READ (14)K
READ (14)(X(J),J=2,6,2)
```

```
      RETURN
      END
```

3. Пусть имеется набор данных последовательного доступа с номером 17, состоящий из одной логической записи, которая содержит числа:

—276.3, 69.7, 32.8, —14.06, 99.5, —4.08, 108.9, —215.4, 68.10, —3.76, 64.27, 15.60, 18.03, 14.01, 0.1, —0.02, —0.03, —4.1, 12.06, 1.9, — 1.8, 1.12, —3.3, 4.8, 4.9, 10.2, 6.7.

Эти числа имеют стандартную длину, относятся к вещественному типу и представлены в машинной форме.

Определить значения элементов массива X и переменной A, которые они получат в результате выполнения следующей программы:

```
      REAL A,X (5,5)
      DO 3 I=1,5
      DO 3 J=1,5
    3 X(I, J)=0.0
      READ (17)((X(J,I),A, I=2,5,3),J=2,5)
      RETURN
      END
```

4. Пусть задана программа:

```
      INTEGER *2JAN(10)/2,8,—4,3*5,—5,2*3,10/
      M=0
      DO 2 I=1,10
    2 M=M+JAN(I)
      WRITE (3) (JAN(I), I=1,10,2)
      WRITE (3) M
      STOP
      END
```

Определить последовательность чисел, которые будут записаны в первую и во вторую логические записи набора 3.

5. Пусть задана программа:

```
      REAL X(5,3)/—0.12,12.1,14.0,3*0.5,—4.1,5.0,3*6.2,2*—4.12,
           10.,9.1/
      WRITE (15) ((X(J,I),J=2,5,2),I=1,3)
      WRITE (15) ((X(J,I),J=1,5,2),I=1,3)
      RETURN
      END
```

Определить последовательность чисел, которые будут помещены в первую и во вторую логические записи набора 15.

6. Пусть задан одномерный массив A(10) целых чисел. Составить программу вывода элементов этого массива в набор данных 23 последовательного доступа, содержащий три логические записи, так, чтобы первая запись содержала элементы массива с номерами 2,4,6,8,10; вторая запись — с номерами 1,3,5,7,9; третья запись — с номерами 1,2,4,5,7,8.

7. Пусть заданы:

а) Набор данных последовательного доступа со ссылочным номером 2, содержащий одну логическую запись, в которой хранятся 35 значений элементов массива M1. Массив M1 — одномерный, относится к целому типу стандартной длины. Значения элементов размещаются в записи в порядке возрастания номеров этих элементов M1(1), M1(2) и т. д.

б) Одномерный массив M2, относящийся к целому типу стандартной длины и содержащий 35 элементов.

Составить программу, обеспечивающую чтение из набора 2 значений элементов массива M1 и присваивание их элементам массива M2 таким образом, чтобы 35-й элемент массива M2 получил значение 1-го элемента массива M1, 34-й элемент массива M2 — значение 2-го элемента массива M1 и т. д.

8. Пусть заданы:

а) Набор данных последовательного доступа со ссылочным номером 3, содержащий две логические записи. Во второй логической записи хранятся 30 значений элементов одномерного массива M1, относящегося к целому типу стандартной длины. Значения элементов размещаются в записи в порядке возрастания номеров этих элементов (M1(1), M1(2) и т. д.).

б) Одномерный массив M2, относящийся к целому типу стандартной длины и содержащий 40 элементов.

Составить программу, обеспечивающую чтение из набора 3 значений элементов массива M1 и присваивание их элементам массива M2 таким образом, чтобы первые 10 элементов массива M2 остались неизменными, 11-й элемент массива M2 получил бы значение 1-го элемента массива M1, 12-й элемент массива M2 — значение 2-го элемента массива M1 и т. д.

9. Пусть заданы:

а) Набор данных последовательного доступа со ссылочным номером 5, содержащий одну логическую запись, в которой хранятся 50 значений элементов массива X. Массив X — одномерный и относится к вещественному типу стандартной длины. Значения элементов располагаются в записи в порядке возрастания их номеров (X(1), X(2), и т. д.).

б) Одномерный массив Y, относящийся к вещественному типу стандартной длины и содержащий 25 элементов.

Составить программу, обеспечивающую чтение из набора 5 значений элементов массива X и присваивание их элементам массива Y таким образом, чтобы 1-й элемент массива Y получил значение 26-го элемента массива X, 2-й элемент массива Y — значение 27-го элемента массива X и т. д.

В случае возникновения ошибки (в процессе чтения данных из набора) программа должна обеспечить выдачу на пульт управления ЭВМ сигнала об ошибке в виде числа 3.

10. Пусть заданы:

а) набор данных последовательного доступа со ссылочным номером 1, содержащий n логических записей ($n \leqslant 50$), каждая из которых хранит по 15 вещественных чисел нестандартной длины;

б) набор данных прямого доступа со ссылочным номером 2, содержащий 50 физических записей длиной по 150 байтов.

Составить программу переписи чисел из набора 1 в набор 2 таким образом, чтобы числа из первой логической записи набора 1 попали в первую физическую запись набора 2, из второй логической записи набора 1 — во вторую физическую запись набора 2 и т. д.

6.2.3. Оператор возврата на первую запись. Оператор возврата на первую запись имеет вид:

n REWIND *a*

где *n* — либо пусто, либо метка, помечающая данный оператор; *a* — ссылочный номер набора данных.

Если набор данных был открыт как выходной, то этот оператор выполняет следующие действия:

1) осуществляет закрытие и записывает признак конца набора данных;

2) устанавливает набор данных на первую логическую запись, то есть делает доступной для операторов вывода и ввода первую (по порядку следования) логическую запись набора данных.

Если набор данных был открыт как входной, то этот оператор выполняет следующие действия:

1) осуществляет закрытие набора данных;

2) устанавливает набор данных на первую логическую запись.

Рассмотрим несколько примеров.

Пример 1. Пусть задана программа:

```
INTEGER A(25)/25*1/,B(25)/25*2/
WRITE (8)A
```

```
      REWIND 8
      WRITE (8)B
      RETURN
      END
```

Эта программа осуществляет запись в набор данных последовательного доступа значений элементов двух массивов: массива A и массива B. Ссылочный номер этого набора — 8.

В результате выполнения оператора WRITE (8)A набор со ссылочным номером 8 будет открыт как выходной, и значения элементов массива A будут помещены в первую логическую запись этого набора. В результате выполнения оператора REWIND 8 набор данных со ссылочным номером 8 будет закрыт и будет установлен на первую запись. В результате выполнения оператора WRITE (8)B набор данных со ссылочным номером 8 будет снова открыт как выходной, и значения элементов массива B будут помещены в его первую логическую запись (на место значений элементов массива A).

Пример 2. Пусть задана программа:

```
      INTEGER A(25)/25*1/,B(25)/25*2/
      WRITE (8)A
      WRITE (8)B
      REWIND 8
      READ (8)B
      RETURN
      END
```

Оператор WRITE (8)A этой программы открывает последовательный набор данных со ссылочным номером 8 как выходной и помещает значения элементов массива A в его первую логическую запись. Оператор WRITE (8)B помещает значения элементов массива B во вторую запись 8-го набора. Оператор REWIND 8 закрывает этот набор и устанавливает его на первую логическую запись. Оператор READ (8)B открывает 8-й набор как входной, читает из первой логической записи этого набора значения элементов массива A и присваивает их элементам массива B.

6.2.4. Оператор возврата на одну запись. Оператор возврата на одну запись обеспечивает возврат набора данных на одну логическую запись назад, т. е. делает доступной операторам ввода и вывода предыдущую логическую запись. Если набор данных установлен на первую запись, то этот оператор не выполняет никаких действий.

Оператор возврата на одну запись имеет вид:

n BACKSPACE *a*

где *п* — либо пусто, либо метка, помечающая данный оператор; *а* — ссылочный номер набора данных.

Оператор BACKSPACE можно выполнить только в том случае, если набор данных *а* находится в открытом состоянии.

Рассмотрим п р и м е р. Пусть задана программа:

```
      REAL *8 A(15)/15 *0.0/,B(5),C(10),D(15)
      A(1) = 0.5
      DO 100 I = 1,9
  100 A(I + 1) = A(I) + 0.5
  200 WRITE (1)A
      DO 101 I = 1,15
  101 A(I) = A(I) + 1.0
  201 WRITE (1)A
      REWIND 1
      READ (1)B
      BACKSPACE 1
      READ (1)C
      READ (1)D
      RETURN
      END
```

Оператор 200 WRITE (1)A этой программы открывает последовательный набор данных со ссылочным номером 1 как выходной, создает в нем первую логическую запись, помещая в нее 10 чисел (0.5, 1.0, . . . , 5.0). Оператор 201 WRITE (1) A создает вторую логическую запись набора, помещая в нее 15 чисел (1.5, 2.0, . . ., 6.0, 1.0, 1.0, . . ., 1.0). Оператор REWIND 1 закрывает набор данных и устанавливает его на первую логическую запись. Оператор READ (1)B открывает этот набор данных как входной, читает числа из его первой логической записи и первые пять чисел присваивает в качестве значений элементам массива B. В результате элементы массива B примут следующие значения: B(1) = 0.5, B(2) = 1.0, . . ., B(5) = = 2.5. Оператор BACKSPACE возвращает набор данных на одну логическую запись назад, в результате чего он будет снова установлен на первую логическую запись. Оператор READ (1)C читает числа из первой логической записи набора и присваивает их в качестве значений элементам массива C, в результате чего они примут следующие значения: C(1) = 0.5, C(2) = 1.0, . . ., C(10) = 5.0. Оператор READ (1)D читает числа из второй логической записи набора и присваивает их в качестве значений элементам массива D, в результате чего они примут следующие значения: D(1) = 1.5, D(2) = 2.0, ..., D(10) = 6.0, D(11) = 1.0, D(12) = 1.0, . . ., D(15) = 1.0.

Глава 7

ФОРМАТЫ ДАННЫХ

Форматы данных — это описания символьных форм представления значений величин в логических записях наборов данных. Эти описания помещаются в операторы форматов и используются операторами форматного ввода-вывода. Перевод данных из символьных форм в машинные осуществляется операторами форматного ввода в процессе их переписи из логических записей в оперативную память. Перевод данных из машинной формы в символьную осуществляется операторами форматного вывода в процессе их переписи из оперативной памяти в логические записи.

7.1. Формат типа I

Формат типа I служит для описания форм представления значений величин целого типа в десятичной системе счисления. Он имеет вид

Iw

где w — целая константа без знака, означающая длину поля (в позициях) логической записи, отведенного под значение величины.

В списке вводимых или выводимых величин этому формату должна соответствовать одна величина целого типа стандартной или нестандартной длины (переменная или элемент массива). Значение этой величины размещается в поле следующим образом:

$$xsa_1a_2 \ldots a_n$$

где x — либо пусто, либо серия пробелов, s — либо пусто, либо знак минус (для отрицательных значений), a_i ($i = 1,2, \ldots, n$) — десятичные цифры, причем a_1 — значащая цифра. Термин «пусто» означает отсутствие символа в позиции для символа.

Для положительных значений позицию для знака числа предусматривать не обязательно. Если количество символов, необходи-

мое для представления значения величины, превышает число w, то при засылке этого значения в логическую запись все позиции поля заполняются символами * (звездочка).

Для описания форм представления m величин используются m форматов.

Рассмотрим несколько примеров.

Пример 1. Пусть форма представления некоторой целой величины описывается форматом I5, а значение этой величины равно 25793. Тогда в логической записи набора данных оно будет представлено в виде последовательности десятичных цифр 25793.

Пример 2. Пусть формы представления значений переменных целого типа X = 152 и Y = —1715 описываются форматами I5 и I6 соответственно. Тогда в логической записи значение X будет представлено последовательностью символов ␣ ␣ 152, а значение Y — последовательностью символов ␣ —1715.

Вопросы и упражнения

1. Для чего предназначен формат типа I?

2. Можно ли форматом типа I описать форму представления в логической записи значения вещественной величины?

3. Чем отличаются форматы, описывающие формы представления значений целых величин стандартной длины и целых величин нестандартной длины?

4. Пусть переменные целого типа имеют следующие значения:

I=1000, J=275, K=—15, M=3100.

Эти значения необходимо поместить в логическую запись набора данных в символьном виде таким образом, чтобы каждое из них занимало поле с минимальным количеством позиций. Описать формы представления значений приведенных выше переменных с помощью форматов типа I.

5. Пусть значения элементов целочисленного массива M1, состоящего из 5-ти элементов, вводятся из логической записи набора данных. Каждое значение занимает в ней поле из 8-ми позиций и представляется в виде десятичного числа таким образом, что последняя цифра числа занимает последнюю позицию соответствующего поля, а знак минус (если число отрицательное) размещается непосредственно перед его первой значащей цифрой. Описать формы представления значений элементов массива M1 в логической записи с помощью форматов типа I.

6. Определить диапазон значений целой величины K, представимой в формате I4.

7.2. Формат типа F

Формат типа F служит для описания форм представления значений величин вещественного и комплексного типов. Он имеет вид

F$w.d$

где w — целая константа без знака, означающая длину поля в логической записи набора, отведенного под значение величины; d — целая константа без знака, означающая количество позиций поля, отведенных под дробную часть значения величины.

В списке вводимых или выводимых величин этому формату должна соответствовать одна величина (переменная или элемент массива вещественного типа, вещественная или мнимая компонента комплексной переменной или элемента комплексного массива). Для описания форм представления m величин используется m форматов.

Значение величины размещается в поле логической записи следующим образом:

$$xsa_1a_2a_3\ldots a_k.b_1b_2\ldots b_d$$

где x — либо пусто, либо серия пробелов, s — либо пусто, либо знак минус (для отрицательных значений); a_i ($i = 1, 2, \ldots, k$) — десятичные цифры целой части величины, причем a_1 — значащая цифра; b_j ($j = 1, 2, \ldots, d$) — десятичные цифры дробной части. Общее количество символов равно w.

Таким образом, значения величин в формате F представляются в том же виде, что и вещественные константы без экспоненты. Если $d = 0$, то значение величины представляется без дробной части, если же $k = 0$, то без целой части. В том и другом случае точка сохраняется. Для положительных значений позицию для знака предусматривать не обязательно.

Если количество символов, необходимое для представления знака, десятичной точки и целой части значений, превышает величину $w - d$, то все позиции поля заполняются символами * (звездочка). Так, например, значение переменной вещественного типа A = —1.0012 в формате F6.3 будет представлено последовательностью символов —1.001, а значение переменной C = 100.5 в этом же формате будет представлено последовательностью символов ******.

Для описания формы представления значений переменной или элемента массива комплексного типа требуется два формата: первый — для вещественной компоненты, второй — для мнимой. Так, например, величина комплексного типа Z = (1.2,3.5) в форматах

F5.1 и F5.1 будет представлена в виде последовательности символов ␣␣1.2␣␣3.5, а в форматах F3.1 и F5.1 — в виде последовательности символов 1.2␣␣3.5.

Вопросы и упражнения

1. Определить диапазон значений величин вещественного типа, представимых в формате F7.2.

2. С какой точностью будут представляться значения величин вещественного типа в формате F6.3?

3. Пусть переменная X вещественного типа принимает значения от —0.01 до 1.3. Написать формат, обеспечивающий представление значений этой переменной в виде десятичных чисел с точностью до 5-ти знаков после десятичной точки.

4. Пусть на перфокарту нанесены значения комплексных переменных A, B и C, которые занимают на ней следующие позиции:

вещественная компонента	A — с 1-й по 10-ю;
мнимая компонента	A — с 11-й по 20-ю;
вещественная компонента	B — с 21-й по 30-ю;
мнимая компонента	B — с 31-й по 40-ю;
вещественная компонента	C — с 41-й по 52-ю;
мнимая компонента	C — с 53-й по 60-ю.

Под дробные части компонент этих переменных отведены по 4 позиции (с 7-й по 10-ю, с 17-й по 20-ю и т. д.). Описать формы представления значений переменных A, B и C форматами типа F.

5. Чем отличаются форматы типа F, предназначенные для описания форм представления значений вещественных величин стандартной и нестандартной длины?

6. Может ли форма представления значения вещественной компоненты комплексной величины описываться форматом типа F, а мнимой — форматом типа I?

7. Могут ли формы представления значений вещественной и мнимой компонент комплексной величины описываться форматами типа F с разными значениями параметра w (ширина поля)?

7.3. Формат типа E

Формат типа E служит для описания форм представления значений величин вещественного типа стандартной длины и значений величин комплексного типа стандартной длины в десятичной системе счисления. Он имеет вид:

Ew.d

где w — целая константа без знака, означающая длину поля в логической записи, отведенного под значение величины; d — целая константа без знака, означающая количество позиций поля, отведенных под дробную часть мантиссы. В списке вводимых или выводимых величин этому формату должна соответствовать одна величина стандартной длины (вещественная переменная, элемент вещественного массива, вещественная или мнимая компонента комплексной переменной, вещественная или мнимая компонента элемента комплексного массива). Для описания форм представления m величин используется m форматов.

Значение величины размещается в поле логической записи следующим образом:

$$xs_1 0.b_1 b_2 \ldots b_d E s_2 a_1 a_2$$

где x — либо пусто, либо серия пробелов, s_1 — либо пусто (для положительных значений), либо знак минус (для отрицательных значений); 0 — цифра нуль, означающая целую часть мантиссы; b_i ($i = 1, 2, \ldots, d$) — цифры дробной части мантиссы, причем для значений, отличных от нуля, $b_1 \neq 0$; E — буква; a_1 и a_2 — цифры десятичного порядка; s_2 — либо пробел (для положительного порядка), либо знак минус (для отрицательного порядка). Общее количество символом равно w. Таким образом, значения величин в формате E представляются в том же виде, что и вещественные константы с экспонентой типа E.

Для формата типа E должно соблюдаться условие $w \geqslant d + 7$. Для описания формы представления значения комплексной переменной требуется два формата: один — для вещественной компоненты, второй — для мнимой.

Рассмотрим несколько примеров.

Пример 1. Пусть задан формат E12.4. Тогда значение вещественной переменной стандартной длины A = —1.001 в этом формате будет представлено в виде последовательности символов

␣—0.1001E␣01

а значение вещественной переменной стандартной длины A = 95.05 в этом же формате будет представлено в виде последовательности символов

␣␣0.9505E␣02

Пример 2. Пусть заданы форматы E9.2 и E12.1 Тогда значение комплексной переменной X = (—1.2, 3.4) в этих двух форма-

тах будет представлено в виде последовательности символов

—0.12E_01_____0.3E_01

В этом примере предполагается, что формат E9.2 относится к вещественной компоненте, а формат E12.1 — к мнимой компоненте переменной X.

Пример 3. Пусть задан формат E11.3. Тогда значение вещественных переменных стандартной длины A = 238.0, B = 0.002 и C = = —21.0057 в этом формате будут представлены последовательностью символов

__0.238E_03__0.200E—02_—0.210E._02

(при условии, что они размещаются в логической записи непосредственно друг за другом в указанном выше порядке).

Вопросы и упражнения

1. Определить диапазон значений величин вещественного типа стандартной длины, представимых в формате E12.4.

2. С какой точностью будут представляться значения величин вещественного типа стандартной длины в формате E12.4?

3. Может ли формат типа E использоваться для описания форм представления значений вещественных и комплексных переменных нестандартной длины?

4. Может ли форма представления значения вещественной компоненты комплексной переменной стандартной длины описываться форматом типа F, а мнимой — форматом типа E?

5. Может ли форма представления значения вещественной компоненты комплексной переменной стандартной длины описываться форматом типа E, а мнимая — форматом типа I?

6. Пусть заданы вещественные переменные стандартной длины A = —76.15, B = 3.001, C = —0.099. Написать строку символов, которая будет представлять совокупность значений этих переменных в формате:

а) E12.4 б) F12.4

Значения переменных A, B и C должны следовать непосредственно друг за другом в указанном выше порядке.

7. Пусть на перфокарте нанесены значения трех вещественных переменных стандартной длины в формате типа E в следующем виде:

__—0.7689E—03_____0.482E—04
__0.1993E_02.

Определить формат представления значения каждой из этих переменных.

7.4. Формат типа D

Формат типа D служит для описания форм представления величин вещественного типа нестандартной длины и величин комплексного типа нестандартной длины в десятичной системе счисления. Он имеет вид:

D$w.d$

где w — целая константа без знака, означающая длину поля, отведенного под значение величины; d — целая константа без знака, означающая количество позиций поля, отведенных под дробную часть мантиссы. В списке вводимых или выводимых величин этому формату должна соответствовать одна величина нестандартной длины (вещественная переменная, элемент вещественного массива, вещественная или мнимая компонента комплексной переменной, вещественная или мнимая компонента элемента комплексного массива).

Для описания форм представления m величин используется m форматов. Значение величины размещается в поле логической записи следующим образом:

$$x s_1 0.b_1 b_2 \ldots b_d \mathrm{D} s_2 a_1 a_2$$

где x — либо пусто, либо серия пробелов; s_1 — либо пусто (для положительных значений), либо знак минус (для отрицательных значений); 0 — цифра нуль, означающая целую часть мантиссы; b_i ($i = 1, 2, \ldots, d$) — цифры дробной части мантиссы, причем для значений, отличных от нуля, $b_1 \neq 0$; D — буква; a_1 и a_2 — цифры десятичного порядка; s_2 — либо пробел (для положительного порядка), либо знак минус (для отрицательного порядка). Общее количество символов равно w.

Таким образом, значения величин в формате D представляются в том же виде, что и вещественные константы с экспонентой типа D.

Для формата типа D должно соблюдаться условие $w \geqslant d + 7$. Для описания формы представления значения комплексной переменной требуется два формата: один для вещественной компоненты, второй — для мнимой.

Рассмотрим два примера.

Пример 1. Пусть задан формат D14.3. Тогда значение вещественной переменной нестандартной длины Y = —14.6 в этом формате будет представлено в виде последовательности символов

____—0.146D_02

Пример 2. Пусть задан формат D10.3. Тогда значения вещественных переменных нестандартной длины A = 238.0, B = = —0.002 и C = —21.0057 в этом формате будут представлены последовательностью символов

_0.238D_03—0.200D—02—0.210D_02

(при условии, что они размещаются в наборе непосредственно друг за другом в указанном выше порядке).

Вопросы и упражнения

1. Определить диапазон значений величин вещественного типа нестандартной длины, представимых в формате D11.2.

2. С какой точностью представляются значения величин вещественного типа нестандартной длины в формате D11.2?

3. Может ли формат типа D использоваться для описания форм представления значений вещественных и комплексных переменных стандартной длины?

4. Может ли форма представления значения вещественной компоненты комплексной переменной нестандартной длины описываться форматом типа F, а мнимой — форматом типа D?

5. Пусть заданы вещественные переменные нестандартной длины A1 = —1643010.95, A2 = — 190000.53, A3 = 99.01, A4 = 36.99, A5 = — 920.15.

Написать строку символов, которая будет представлять совокупность значений этих переменных в формате D12.4. Значения переменных должны размещаться непосредственно друг за другом в указанном выше порядке.

6. Пусть в некотором наборе данных содержится строка символов

__—0.918D_03__0.755D—01__0.11D_08

представляющая собой последовательность значений переменных вещественного типа нестандартной длины в формате типа D.

Определить количество этих переменных и написать их форматы.

7.5. Особенности представления значений величин, предназначенных для ввода в форматах I, F, E и D

Рассмотренные нами выше способы представления величин в форматах I, F, E и D являются основными. Они используются как при выводе значений величин на внешние носители информации, так

и при их вводе с внешних носителей информации. Однако для ввода значения величин могут представляться в несколько ином виде. Рассмотрим эти случаи.

1. Цифровые части значений величин вместо нулей могут содержать пробелы. Так, например, число —208 в формате I может быть представлено в следующем виде: ⎵—2⎵8

2. Буква E в десятичной экспоненте может быть опущена. В этом случае положительный порядок должен представляться со знаком + (плюс), а отрицательный со знаком —(минус). Так, например, число 3.14 в формате типа E может быть представлено в следующем виде: ⎵0.314 + 01.

3. Десятичный порядок может содержать одну цифру. Так, например, число 0.0314 в формате типа D может быть представлено в следующем виде: 0.314D — 1.

4. В значениях величин, вводимых по форматам типа F, E и D, десятичная точка может быть опущена. В этом случае ее положение будет определяться по величине d, указанной в формате.

5. В значениях величин, вводимых по форматам типа F, E и D положение десятичной точки может не соответствовать величине d, указанной в формате. В этом случае положение десятичной точки будет считаться таким, каким оно указано на носителе информации.

6. Если в десятичной экспоненте буква E или D присутствует, то под положительный знак порядка позицию можно не отводить. Так, например, число 3.14 в формате типа E может быть представлено в следующем виде: ⎵0.314E1.

7.6. Формат типа L

Формат типа L служит для описания символьных форм представления значений величин логического типа. Он имеет вид:

Lw

где w — целая константа без знака, означающая длину поля, отведенного под значение величины.

Значение логической величины размещается в поле следующим образом:

xa

где x — либо пусто, либо серия пробелов, a — либо буква T (когда величина имеет значение истина), либо буква F (когда величина имеет значение ложь). Общее количество символов равно w.

Так, например, значение переменной A = .FALSE. в формате L2 будет представлено в виде последовательности символов _F, а значения переменных B=.FALSE. и C=.TRUE. в формате L4 будут представлены в виде последовательности символов _ _ _F и последовательности символов _ _ _T.

Значения логических переменных, предназначенные для ввода, могут представляться в дополнительных формах, имеющих вид:

$$s_1 s_2 \ldots s_n a c_1 c_2 \ldots c_k$$

где s_i ($i = 1, 2, \ldots, n$) — символы, отличные от букв T и F; a — либо буква T (когда величина имеет значение истина), либо буква F (когда величина имеет значение ложь); c_j ($j = 1, 2, \ldots, k$) — произвольные символы.

Значение ложь, предназначенное для ввода, может быть представлено пробелами. Количество пробелов должно быть равно w.

Вопросы и упражнения

1. Пусть задан формат L3. Написать основную и 4 дополнительные формы представления в этом формате значения логической переменной: а) X = .TRUE. б) Y= .FALSE.

2. Чем отличаются форматы типа L, предназначенные для описания форм представления значений логических переменных стандартной и нестандартной длины?

3. Пусть в некотором наборе данных содержится строка символов _ _ TRU _ _ F _ _ TT, представляющая собой последовательность значений трех логических переменных нестандартной длины в формате типа L. Написать этот формат.

7.7. Формат типа A

Формат типа A служит для описания формы представления значения величины любого типа в виде последовательности произвольных символов. Он имеет вид:

Aw

где w — целая константа без знака, означающая длину поля, отведенного под значение величины. В списке вводимых или выводимых величин этому формату должна соответствовать одна величина.

Считается, что значениями этой величины могут быть последовательности символов, которые в основной памяти представляются в виде восьмиразрядных двоичных кодов.

Количество k символов, образующих значение величины, берется равным количеству байтов памяти, отводимому под значение данной величины. Так, например, для переменной целого типа нестандартной длины $k = 2$, для переменной целого типа стандартной длины $k = 4$ и т. д.

Рассмотрим правила заполнения полей при выводе значений переменных и правила формирования значений переменных при их вводе.

1. Пусть $w = k$. Если поле на внешнем носителе информации заполнено символами $s_1 s_2 \ldots s_w$, то при вводе соответствующая переменная получит символьное значение $s_1 s_2 \ldots s_w$. Если переменная имеет символьное значение $s_1 s_2 \ldots s_k$, то при выводе соответствующее поле на внешнем носителе информации будет заполнено символами $s_1 s_2 \ldots s_k$.

2. Пусть $w > k$. Если поле на внешнем носителе информации заполнено символами $s_1 s_2 \ldots s_w$, то при вводе соответствующая переменная получит символьное значение $s_{w-k+1} s_{w-k+2} \ldots s_w$ (левые символы поля отсекаются). Если переменная имеет символьное значение $s_1 s_2 \ldots s_k$, то при выводе соответствующее поле на внешнем носителе информации будет заполнено символами $\_\_\ldots\_ s_1 s_2 \ldots s_k$ (дополнится слева пробелами).

3. Пусть $w < k$. Если поле на внешнем носителе информации заполнено символами $s_1 s_2 \ldots s_w$, то при вводе соответствующая переменная получит символьное значение $s_1 s_2 \ldots s_w \_\_\ldots\_$ (дополнится справа пробелами). Если переменная имеет символьное значение $s_1 s_2 \ldots s_k$, то при выводе соответствующее поле на внешнем носителе информации будет заполнено символами $s_1 s_2 \ldots s_w$ (правые символы отсекаются).

Рассмотрим несколько примеров.

Пример 1. Пусть в наборе данных содержится строка символов SINX. Будем считать, что эта строка является значением некоторой переменной, представленной в формате A4 и с помощью оператора форматного ввода вводится в оперативную память и присваивается в качестве значения переменной A. Тогда переменная A получит следующее символьное значение:

а) последовательность символов SINX, если A относится к целому типу стандартной длины;

б) последовательность символов SINX____, если A относится к вещественному типу нестандартной длины;

в) последовательность символов NX, если A относится к целому типу нестандартной длины.

Пример 2. Пусть переменная вещественного типа стандартной длины имеет символьное значение, равное DATA. Тогда:

а) при выводе значения этой переменной в формате A4, оно запишется в набор данных в виде строки символов DATA;

б) при выводе значения этой переменной в формате A6, оно запишется в набор данных в виде строки символов __DATA;

в) при выводе значения этой переменной в формате A3, оно запишется в набор данных в виде строки символов DAT.

Вопросы и упражнения

1. Для каких целей можно использовать в фортран-программах форматы типа A?

2. Пусть в некоторой программе содержатся следующие описания типов:

```
LOGICAL A/4HXYZW/,B*1/'*'/
REAL *8 E/'12345678'/
```

Значения переменных A, B и E в программе не изменяются и выдаются на печать (в набор данных на бумажной ленте) в формате A4. Написать строки символов, которые будут выданы на печать в качестве значений переменных A,B и E.

3. Пусть в наборе данных содержится строка символов 8МАРТА. Будем считать, что эта строка является значением некоторой переменной, представленным в формате A6, и с помощью оператора форматного ввода вводится в оперативную память и присваивается в качестве значения переменной A.

Определить символьное значение переменной A, если:

а) переменная A относится к логическому типу стандартной длины;

б) переменная A относится к вещественному типу стандартной длины;

в) переменная A относится к вещественному типу нестандартной длины.

7.8. Формат типа Z

Формат типа Z служит для описания формы представления значения величины любого типа в виде последовательности шестнадцатеричных цифр. Он имеет вид:

Z*w*

где *w* — целая константа без знака, означающая длину поля, отведенного под значение величины.

В списке вводимых или выводимых величин этому формату должна соответствовать одна величина. В основной памяти шестнадцатеричные цифры представляются в виде четырехразрядных двоичных кодов.

На внешних носителях информации шестнадцатеричные цифры представляются таким же образом, как и другие символы.

Количество k шестнадцатеричных цифр, образующих значение величины, берется равным удвоенному количеству байтов памяти, отводимых под значение данной величины. Так, например, для переменной целого типа нестандартной длины $k = 4$, для переменной целого типа стандартной длины $k = 8$ и т. д.

Рассмотрим правила заполнения полей при выводе значений переменных и правила формирования значений переменных при их вводе.

1. Пусть $w = k$. Если поле на внешнем носителе информации заполнено шестнадцатеричными цифрами $s_1 s_2 \ldots s_w$, то при вводе соответствующая переменная получит шестнадцатеричное значение $s_1 s_2 \ldots s_k$. Если переменная имеет шестнадцатеричное значение $s_1 s_2 \ldots s_k$, то при выводе соответствующее поле на внешнем носителе информации будет заполнено символами $s_1 s_2 \ldots s_k$.

2. Пусть $w > k$. Если поле на внешнем носителе информации заполнено шестнадцатеричными цифрами $s_1 s_2 \ldots s_w$, то при вводе соответствующая переменная получит шестнадцатеричное значение $s_{w-k+1} s_{w-k+2} \ldots s_w$ (левые символы поля отсекаются). Если переменная имеет шестнадцатеричное значение $s_1 s_2 \ldots s_k$, то при выводе соответствующее поле на внешнем носителе информации заполнится символами $\_\_\ldots\_ s_1 s_2 \ldots s_k$ (дополнится слева пробелами).

3. Пусть $w < k$. Если поле на внешнем носителе информации заполнено шестнадцатеричными цифрами $s_1 s_2 \ldots s_w$, то при вводе соответствующая переменная получит шестнадцатеричное значение $0 \ldots 0\, s_1 s_2 \ldots s_k$ (дополнится слева нулями). Если переменная имеет шестнадцатеричное значение $s_1 s_2 \ldots s_k$, то при выводе соответствующее поле на внешнем носителе информации заполнится символами $s_{k-w+1} s_{k-w+2} \ldots s_k$ (левые символы отсекаются). При вводе по формату Z пробел воспринимается как шестнадцатеричная цифра нуль.

Рассмотрим несколько примеров.

Пример 1. Пусть в наборе данных содержится строка символов $\_\_19$. Будем считать, что эта строка является значением некоторой переменной, представленным в формате Z4, и с помощью оператора форматного ввода вводится в оперативную память и при-

сваивается в качестве значения переменной А. Тогда переменная А получит следующее шестнадцатеричное значение:

а) шестнадцатеричное число 0019, если переменная А относится к целому типу нестандартной длины;

б) шестнадцатеричное число 19, если переменная А относится к логическому типу нестандартной длины;

в) шестнадцатеричное число 00000019, если переменная А относится к целому типу стандартной длины.

Пример 2. Пусть переменная I целого типа нестандартной длины имеет значение, равное 1B2D, и выводится в набор данных с помощью оператора форматного вывода. Тогда значение переменной I запишется в набор данных в виде последовательности символов:

а) 1B2D, если вывод осуществлялся в формате Z4;

б) __ __ 1B2D, если вывод осуществлялся в формате Z6;

с) 2D, если вывод осуществлялся в формате Z2.

Вопросы и упражнения

1. Для каких целей можно использовать в фортран-программах форматы типа Z?

2. Пусть переменная относится к вещественному типу, имеет стандартную длину и ее значение, представленное в виде последовательности шестнадцатеричных цифр, имеет вид:

1234EFCD

Написать последовательность символов, которая будет представлять это значение:

а) в формате Z2;

б) в формате Z8;

в) в формате Z10.

3. Пусть в наборе данных содержится строка символов 01ACDF00, которая является значением некоторой переменной, представленным в формате Z8, и с помощью оператора форматного ввода вводится в оперативную память и присваивается переменной А. Определить шестнадцатеричное значение переменной А, если:

а) переменная А относится к целому типу и имеет стандартную длину;

б) переменная А относится к целому типу и имеет нестандартную длину;

в) переменная А относится к вещественному типу и имеет нестандартную длину.

7.9. Символьный формат

Символьный формат имеет вид:

C

где *C* — символьная константа, представленная либо с указателем длины, либо без указателя длины. Длина поля на внешнем носителе информации определяется количеством символов, содержащихся в константе (указатель длины и внешние апострофы в счет не берутся).

Так, например, константа ТАБЛИЦА в символьном формате может быть представлена двумя способами:

1) 'ТАБЛИЦА'
2) 7Н ТАБЛИЦА

В списке вводимых или выводимых величин этому формату ничего не соответствует. При выводе символы, содержащиеся в константе *C*, помещаются в соответствующее поле набора данных. При вводе символы, содержащиеся в соответствующем поле набора данных, помещаются на место символов константы *C*, содержащейся в формате.

7.10. Формат типа X

Формат типа X имеет вид:

*w*X

где *w* — целая константа без знака, означающая длину поля в логической записи набора данных. В списке вводимых или выводимых величин этому формату ничего не соответствует.

При вводе этот формат вызывает пропуск поля длиной в *w* позиций. При выводе все позиции поля заполняются пробелами. Длина поля *w* не должна превосходить 255 позиций.

7.11. Формат типа T

Формат типа T служит для указания порядкового номера позиции в логической записи, с которой начинается последующий ввод или вывод данных. Он имеет вид:

T*w*

где *w* — целая константа без знака, означающая порядковый номер позиции в логической записи.

Если в процессе вывода данных в позиции с k-й ($k < w$) по (w — 1)-ю ничего не засылалось, то формат типа T вызовет заполнение этих позиций пробелами.

7.12. Формат типа G

Формат типа G является универсальным и имеет вид:

G$w.d$

где w — целая константа без знака, означающая длину поля в логической записи, d — целая константа без знака, назначение которой будет раскрыто в процессе дальнейшего описания формата.

В списке вводимых или выводимых величин этому формату должна соответствовать переменная или элемент массива целого, вещественного или логического типа. Кроме того, этому формату может соответствовать одна компонента (вещественная или мнимая) комплексной величины.

Рассмотрим действие этого формата.

1. Пусть формату G$w.d$ соответствует величина целого типа стандартной или нестандартной длины. Тогда действие этого формата будет эквивалентно действию формата Iw. Параметр d в этом случае игнорируется и может быть опущен.

2. Пусть формату G$w.d$ соответствует величина логического типа стандартной или нестандартной длины. Тогда действие этого формата будет эквивалентно действию формата Lw. Параметр d в этом случае игнорируется и может быть опущен.

3. Пусть формату G$w.d$ соответствует величина вещественного типа или компонента (вещественная или мнимая) комплексной величины стандартной или нестандартной длины. Тогда при выводе форма представления значения этой величины будет являться функцией значения этой величины. Эта функция для значений $0.1 \leqslant N < 10^d$ приведена в табл. 7.1.

В остальных случаях действие формата G$w.d$ эквивалентно действию формата E$w.d$ (для величин стандартной длины) и формата D$w.d$ (для величин нестандартной длины).

При вводе значений величин вещественного или комплексного типа стандартной длины действие формата G$w.d$ эквивалентно действию формата F$w.d$, или же формата E$w.d$. Это означает, что в логической записи значения, соответствующие формату G$w.d$, могут представляться либо в формате F$w.d$, либо в формате E$w.d$. При вводе значений величин вещественного или комплексного типа нестандартной длины действие формата G$w.d$ эквивалентно действию

Таблица 7.1. Формы представления величины N в формате типа G

Значение величины N	Форматное выражение, эквивалентное формату G$w.d$	Обозначения
$0.1 \leqslant N < 1$	F$k.m_1$, 4X	$k = w - 4$, $m_1 = d$
$1 \leqslant N < 10$	F$k.m_2$, 4X	$k = w - 4$, $m_2 = d - 1$
$10 \leqslant N < 100$	F$k.m_3$, 4X	$k = w - 4$, $m_3 = d - 2$
. . .	. . .	. . .
$10^{d-1} \leqslant N < 10^d$	Fk.0, 4X	$k = w - 4$

формата F$w.d$, или же формата D$w.d$. Это означает, что в логической записи значения, соответствующие формату G$w.d$, могут представляться либо в формате F$w.d$, либо в формате D$w.d$.

Рассмотрим несколько примеров.

Пример 1. Пусть задана целая переменная K = 152. Тогда в форматах G4.3 и G4 значение этой переменной будет представлено в виде строки символов _152.

Пример 2. Пусть задан формат G10.3 и в списке вводимых величин ему соответствует вещественная переменная стандартной длины X = 12.3. В поле логической записи набора данных значение этой переменной может представляться в виде последовательности символов _____12.300, а также в виде последовательности символов _0.123E_02.

Пример 3. Пусть задан формат G10.3 и в списке вводимых величин ему соответствует вещественная переменная нестандартной длины X = 12.3. В поле логической записи набора данных значение этой переменной может представляться в виде последовательности символов _____12.300, а также в виде последовательности символов _0.123D_02.

Пример 4. Пусть вещественная переменная стандартной длины имеет значение 0.57. Тогда при выводе в формате G10.3 это значение будет представлено в виде символов _0.570_____.

Пример 5. Пусть комплексная переменная стандартной длины имеет значение (2.51, 1.738). Тогда при выводе в формате G8.2 вещественная компонента этой переменной будет представлена в виде строки символов _2.5____, а мнимая — в виде строки символов _1.7_____.

Пример 6. Пусть заданы вещественные переменные стандартной длины X = 741.93 и Y = 18.91, а также логическая пере-

менная L = .TRUE. . Тогда при выводе в формате G9.2 значение переменной X будет представлено в виде строки символов _0.74E_03, переменной Y — в виде строки символов __18._____, переменной L — в виде строки символов _______T.

Вопросы и упражнения

1. Для каких целей можно использовать форматы типа G?

2. Может ли форма представления значения вещественной компоненты комплексной переменной описываться форматом типа G, а мнимой компоненты — форматом типа F?

3. Пусть заданы элементы одномерного массива A, относящиеся к вещественному типу и имеющие стандартную длину:

A(1)=0.015939, A(2)=1.5939, A(3)=15.939, A(4)=159.39, A(5)=
=1593.9

Какой вид будут иметь эти значения в формате G10.3?

4. Пусть заданы вещественная переменная стандартной длины A = 0.05555; целая переменная нестандартной длины K = —21; логическая переменная стандартной длины L = .FALSE.. Какой вид будут иметь эти значения в формате G12.4?

5. Пусть числовое значение некоторой переменной X в поле логической записи представлено в виде последовательности символов:

а) _125

б) —135.840

в) __0.123E_02

г) _0.958D—04

Определить, какой тип и какую длину может иметь переменная X в каждом из приведенных выше случаев. Каким форматом можно воспользоваться для ввода этих значений?

7.13. Оператор формата

Как уже указывалось, обмен данными между оперативной памятью и набором данных осуществляется порциями, называемыми логическими записями. Оператор форматного вывода при однократном своем выполнении может поместить в набор данных одну или несколько логических записей. Совокупность значений величин, помещаемых в эти записи, задается с помощью списка вводимых величин, входящего в состав оператора вывода. Формы же представления значений величин в логических записях, а также деление всей совокупности значений на логические записи, задаются с по-

мощью оператора формата, связанного с данным оператором вывода.

Оператор форматного ввода при однократном своем выполнении может прочитать из набора данных также одну или несколько логических записей. Общий объем читаемых из набора данных значений величин определяется списком вводимых величин. Формы же представления значений этих величин в логических записях, а также их деление на логические записи задаются с помощью оператора формата, связанного с данным оператором ввода.

Оператор формата имеет вид:

n FORMAT (*a*)

где *n* — метка, помечающая данный оператор, *a* — форматное выражение.

В простейшем случае форматное выражение представляет собой список форматов, в котором форматы, относящиеся к одной логической записи, отделяются друг от друга запятыми, а группы форматов, относящиеся к различным логическим записям, отделяются друг от друга символом / (косая черта). Таким образом, оператор формата может иметь вид:

n FORMAT(f_1, f_2, f_3 /f_4, f_5/f_6, f_7, f_8, f_9)

где f_1, f_2, f_3 — группа форматов, относящаяся к первой логической записи; f_4, f_5 — группа форматов, относящаяся ко второй логической записи; f_6, f_7, f_8, f_9 — группа форматов, относящаяся к третьей логической записи.

Порядок следования форматов в форматном выражении должен соответствовать:

а) порядку следования описываемых ими полей внутри логических записей и порядку следования логических записей внутри набора данных;

б) порядку следования соответствующих им величин в списке вводимых или выводимых величин.

Рассмотрим несколько примеров.

Пример 1. Пусть задан оператор формата 5 FORMAT (I3, F15.7, L4), который используется оператором вывода. Приведенный выше оператор формата описывает одну логическую запись, состоящую из трех полей. В процессе выполнения оператора вывода будут выполняться следующие действия:

1) значение первой величины, входящей в список выводимых величин, преобразуется из машинной формы в формат I3 и помещается в первое поле записи, занимающее в ней три позиции;

2) значение второй величины, входящей в список выводимых величин, преобразуется из машинной формы в формат F15.7 и помещается во второе поле записи, занимающее в ней пятнадцать позиций;

3) значение третьей величины, входящей в список выводимых величин, преобразуется из машинной формы в формат L4 и помещается в третье поле записи, занимающее в ней четыре позиции.

Пример 2. Пусть задан оператор формата 3 FORMAT (I5, E14.4/F10.5/L4, L5), который используется оператором вывода. Оператор формата описывает три логические записи, первая из которых состоит из двух полей, вторая — из одного поля и третья — из двух.

В процессе выполнения оператора вывода будут выполняться следующие действия:

1) значения первых двух величин, входящих в список выводимых величин, преобразуются из машинной формы в форматы I5 и E14.4 и помещаются в первое и второе поля первой логической записи;

2) значение третьей величины, входящее в список выводимых величин, преобразуется из машинной формы в формат F10.5 и помещается во вторую логическую запись;

3) значения четвертой и пятой величин, входящих в список выводимых величин, преобразуются из машинной формы в форматы L4 и L5 и помещаются в первое и второе поля третьей логической записи.

Если в процессе ввода или вывода требуется пропустить логическую запись, то в форматное выражение для этой записи помещается только разделитель /(косая черта). Так, например, оператор формата, используемый оператором ввода, может иметь вид:

```
15 FORMAT (/I3,I5//I6).
```

Этот оператор содержит описания 4-х логических записей, которые говорят о том, что:

а) первая логическая запись пропускается;

б) из второй логической записи выбираются два значения целых величин, которые хранятся в ней в форматах I3 и I5. Эти значения преобразуются в машинную форму и присваиваются первым двум величинам, входящим в список вводимых величин;

в) третья логическая запись пропускается;

г) из четвертой логической записи выбирается одно значение целой величины, которое хранится в ней в формате I6, переводится

в машинную форму и присваивается третьей величине, входящей в список вводимых величин.

Рассмотрим прием, позволяющий записывать форматные выражения в компактной форме. Для этого введем следующие понятия:

а) *первичный формат* — это формат типа I,F,E,D,G,L,A,Z;

б) *список форматов* — это последовательность форматов, разделенных запятыми.

Прием компактной записи заключается в следующем:

1) Пусть форматное выражение содержит список первичных форматов $f_1, f_2, \ldots, f_n$, причем $f_1 = f_2 = \ldots = f_n = f$. Тогда такой список можно заменить на выражения вида nf, где n — целое число без знака, называемое *повторителем* и означающее количество форматов в приведенном выше списке. Например, список форматов 15,15,15,15 можно представить в виде выражения 4I5.

Выражение вида nf принято называть *форматом группы 1-го уровня*.

2) Пусть форматное выражение содержит список $c_1, c_2, \ldots, c_n$, где c_i ($i = 1, 2, \ldots, n$) — списки первичных форматов, форматов типа X, T, символьных форматов и форматов группы первого уровня, причем $c_1 = c_2 = \ldots = c_n = c$. Тогда такой список можно заменить на выражение вида $n(c)$, где n — повторитель. Например, список форматов I5, F7.1,L5, I5,F7.1,L5 можно представить в виде выражения 2(I5, F7.1,L5).

Заметим, что равенство $c_i = c_j$ означает равенство числа элементов в списках c_i и c_j, а также попарное равенство их соответствующих элементов. Выражение $n(c)$ принято называть *форматом группы 2-го уровня*.

3) Пусть форматное выражение содержит список $s_1, s_2, \ldots, s_n$, где s_i ($i = 1, 2, \ldots, n$) — списки первичных форматов, форматов групп первого уровня и форматов групп второго уровня, причем $s_1 = s_2 = \ldots = s_n = s$. Тогда такой список можно заменить на выражение вида $n(s)$, где n — повторитель. Например, список форматов 5F7.1,3(I5, L7), 5F7.1, 3(I5, L7) можно представить в виде выражения 2(5F7.1,3(I5, L7)). Выражение вида $n(s)$ принято называть *форматом группы 3-го уровня*.

Таким образом, общий вид форматного выражения можно представить так:

$$(\alpha_1 f_1 \beta_1 f_2 \beta_2 \ldots \beta_{n-1} f_n \alpha_2)$$

где f_i ($i = 1, 2, \ldots, n$) — форматы (первичные форматы, а также форматы группы 1, 2 и 3 уровней); β ($i = 1, \ldots, n$) — либо запя-

тая, либо серия косых черточек; α_i ($i = 1, 2$) — либо пусто, либо серия косых черточек.

Приведем в качестве примеров следующие операторы форматов:

5 FORMAT(I3,4X,3F10.4)

3 FORMAT(I2,5E14.4/8F10.3)

7 FORMAT (//I5.13,2(2X,3F6.2)/7L4)

Теперь рассмотрим более детально взаимосвязь оператора формата с операторами ввода и вывода.

1. Пусть оператор формата используется совместно с оператором вывода. В этом случае вывод начинается с перебора форматов, которые перебираются слева направо с учетом повторителей и скобок. Если формат требует обращения к списку выводимых величин (например, формат типа I), то из этого списка выбирается очередная величина. Значение этой величины преобразуется в форму, соответствующую формату, и засылается в поле записи. Если формат не требует обращения к списку выводимых величин (например, формат типа X), то выполняются необходимые действия без обращения к списку. Величина, значение которой было помещено в запись, становится недоступной для следующей выборки. При повторном обращении к списку выводимых величин будет выбираться следующая величина.

Если список выводимых величин будет исчерпан раньше, чем форматное выражение, то перебор форматов будет продолжаться до тех пор, пока не встретится формат, требующий обращения к списку выводимых величин. После этого выполнение оператора вывода будет прекращено.

Если же форматное выражение будет исчерпано раньше, чем список выводимых величин, то перебор форматов будет повторяться, начиная с *точки повторения форматов*. Под точкой повторения форматов понимается косая черта или формат (первичный или формат группы), с которого начинается перебор элементов форматного выражения после его исчерпания. Опишем способ определения этой точки.

Пусть форматное выражение имеет вид:

$$(\alpha_1 f_1 \beta_1 f_2 \beta_2 \ldots \beta_{t-1} f_t \beta_t \ldots \beta_{n-1} f_n \alpha_2).$$

Если формат f_t является форматом группы 2-го или 3-го уровня, а форматы $f_{t+1}, f_{t+2}, \ldots, f_n$ относятся к первичным форматам или к форматам группы 1-го уровня, то повторный перебор форматов начнется с формата f_t. Если же форматы $f_1, f_2, \ldots, f_n$ относятся к первичным форматам или к форматам группы первого уровня, то повторный перебор форматов начнется с начала форматного выражения.

Каждый переход через косую черту, а также переход на повторный перебор форматов (после его очередного исчерпания) сопровождается переходом на следующую логическую запись набора. В этом случае выполнение оператора вывода прекращается только тогда, когда будет исчерпан весь список выводимых величин.

2. Пусть оператор формата используется совместно с оператором ввода. В этом случае ввод начинается с перебора форматов, которые перебираются слева направо с учетом повторителей и скобок. Если формат требует обращения к списку вводимых величин (например, формат типа I), то из очередного поля логической записи выбирается значение величины, которое переводится в машинную форму и присваивается очередной величине, выбранной из списка вводимых величин. Если же формат не требует обращения к списку вводимых величин (например, формат типа X), то выполняются необходимые действия без обращения в списку. Величина, которая была выбрана из списка, и которой было присвоено значение, становится недоступной для следующей выборки. При повторном обращении к списку вводимых величин будет выбираться следующая величина.

Если список вводимых величин будет исчерпан раньше, чем форматное выражение, то перебор форматов будет прекращен лишь тогда, когда встретится формат, требующий обращения к списку вводимых величин.

Если же форматное выражение будет исчерпано раньше, чем список вводимых величин, то перебор форматов будет повторяться начиная с точки повторения форматов.

Каждый переход через косую черту и переход на повторный перебор форматов (после его очередного исчерпания) сопровождается переходом на следующую логическую запись набора. В этом случае выполнение оператора ввода прекращается только тогда, когда будет исчерпан весь список вводимых величин.

Вопросы и упражнения

1. Пусть задан список первичных форматов E7.2, E7.2, E7.2. Представить его в компактном виде (в виде формата группы первого уровня).

2. Пусть задан формат группы второго уровня 3(I5, 2L4). Представить его в виде списка первичных форматов.

3. Может ли оператор формата 5 FORMAT(I7,I15) использоваться оператором вывода для помещения в одну логическую запись набора данных:

а) значений двух величин целого типа;

б) значения одной величины целого типа;

в) значений трех величин целого типа;

г) значений двух величин вещественного типа?

4. Пусть оператор формата 5 FORMAT(I7,I15) используется оператором вывода для помещения в набор данных значений 20-ти величин целого типа. Сколько логических записей займут эти величины в наборе данных?

5. Пусть оператор формата 7 FORMAT (L3,5(L3,L4)) используется оператором ввода для выборки из набора данных значений 32-х величин логического типа. Сколько логических записей набора данных будет использовано в процессе выполнения оператора ввода?

6. Определить точку повторения форматов в следующих форматных выражениях:

а) (I5, I3, 3I6, I7)

б) (I5,I3,3(I6), I7)

в) (I5, I6/2(I7,I4)/I3)

г) (I5,I6/2(I4)/2(I7,3(I4)))

7. Всегда ли формат группы первого уровня 3I5 равносилен последовательности форматов I5,I5,I5?

8. Всегда ли формат группы второго уровня 2(I4,I5) равносилен последовательности форматов I4,I5,I4,I5?

7.14. Масштабный множитель

Масштабный множитель используется совместно с форматами типа F, E, D и G. Он имеет вид:

*n*P

где *n* — целое число со знаком или без знака, которое может изменяться от —127 до +127; P — указатель масштабного множителя.

Масштабный множитель может указываться либо перед первичными форматами, либо перед форматами группы первого уровня. Так, например, возможны такие сочетания масштабных множителей и форматов:

5PF7.2, —2PE10.3, 4PG15.3, 5P4F7.2,— 3P8G18.1.

Действие масштабного множителя зависит от формата, с которым он используется, и от того, осуществляется ввод данных или их вывод. Рассмотрим эти действия.

1. *Осуществляется ввод.* Используется формат *n*PF*w.d*. В этом случае значение величины, выбранное из логической записи, после перевода в машинную форму будет умножено на 10^{-n}.

2. *Осуществляется ввод.* Используются форматы *n*PE*w.d* и *n*PD*w.d*. В этом случае масштабный множитель игнорируется.

3. *Осуществляется ввод.* Используется формат *n*PG*w.d*. Если значение величины представлено в формате E или формате D, то действие масштабного множителя игнорируется. Если же значение величины представлено в формате F, то значение величины, выбранное из логической записи, после перевода в машинную форму умножается на 10^{-n}.

4. *Осуществляется вывод.* Используется формат *n*PF*w.d*. В этом случае значение величины умножается на 10^n, после чего представляется в формате F*w.d*.

5. *Осуществляется вывод.* Используются форматы *n*PE*w.d* и *n*PD*w.d*. В этом случае после перевода значения величины в формат E*w.d* или в формат D*w.d* десятичная точка в нем смещается на *n* позиций либо вправо (если $n \geqslant 0$), либо влево (если $n < 0$), а порядок корректируется таким образом, чтобы значение величины не изменилось. При этом общее количество позиций (*w*) и число позиций для дробной части мантиссы (*d*) остается прежним.

6. *Осуществляется вывод.* Используется формат *n*PG*w.d*. Значение величины заключено в интервале между 0.1 и 10^d. В этом случае масштабный множитель игнорируется.

7. *Осуществляется вывод.* Используется формат *n*PG*w.d*. Величина имеет стандартную длину, и ее значение не находится в интервале между 0.1 и 10^d. В этом случае значение величины переводится в формат E*w.d*, после чего десятичная точка в нем смещается на *n* позиций либо вправо (если $n \geqslant 0$), либо влево (если $n < 0$), а порядок корректируется так, чтобы значение величины не изменилось.

8. *Осуществляется вывод.* Используется формат *n*PG*w.d*. Величина имеет нестандартную длину, и ее значение не находится в интервале между 0.1 и 10^d. В этом случае значение величины переводится в формат D*w.d*, после чего десятичная точка в нем смещается на *n* позиций либо вправо (если $n \geqslant 0$), либо влево (если $n < 0$), а порядок корректируется так, чтобы значение величины не изменилось.

Приведем несколько примеров.

Пример 1. Осуществляется ввод значения вещественной переменной X. Используется формат 2PF4.2. В логической записи значение X представлено в виде последовательности символов 1.55. После ввода X будет иметь значение, равное 0.0155.

Пример 2. Осуществляется ввод значения вещественной переменной X, имеющей нестандартную длину. Используется формат 2PE10.2. В логической записи значение X представлено в виде по-

следовательности символов __0.25E__03. После ввода X будет иметь значение, равное 250.

Пример 3. Осуществляется вывод значения вещественной величины X = 125.734. Используется формат — 2PF8.3. В логическую запись будет помещена последовательность символов ___1.257

Пример 4. Осуществляется вывод значения вещественной величины X = 1257.34, имеющей стандартную длину. Используется формат 3PE16.5. В логическую запись будет помещена последовательность символов ___125.73400E__01.

Действие масштабного множителя, указанного в одном из форматов, распространяется на все последующие форматы форматного выражения до появления нового масштабного множителя. Для устранения действия предыдущего масштабного множителя можно указать масштабный множитель 0P.

Вопросы и упражнения

1. Для каких целей используется масштабный множитель?

2. На какие форматы распространяется действие масштабного множителя?

3. Пусть задана вещественная переменная стандартной длины X = 14.385, значение которой выводится в логическую запись набора данных. Написать последовательность символов, которая будет представлять значение этой переменной в логической записи, если вывод осуществляется в формате:

а) F8.3; б) 2PF8.3; в) E13.5;

г) —3PE13.5; д) G13.5; е) 4PG13.5.

4. Пусть в логической записи набора данных содержится некоторое вещественное число, которое вводится в оперативную память и присваивается вещественной переменной X, имеющей нестандартную длину. Какое значение будет иметь переменная после ввода, если:

а) в логической записи число представлено последовательностью символов __—31105.007 и ввод осуществляется в формате F11.3;

б) в логической записи число представлено последовательностью символов __—31105.007 и ввод осуществляется в формате 1PF11.3;

в) в логической записи число представлено в виде последовательности символов __—31105.007 и ввод осуществляется в формате —1PF11.3;

г) в логической записи число представлено последовательностью символов —0.31105007D__05 и ввод осуществляется в формате 1PD15.8;

Оператор описания наборов данных и оператор поиска записи выполняют те же функции и имеют тот же вид, что и соответствующие операторы бесформатного ввода-вывода. Исключением является лишь то обстоятельство, что в операторе описания наборов данных, относящемся к группе операторов форматного ввода, в качестве указателя единицы измерения длины физической записи используются символы L и E.

Символ L означает, что физические записи измеряются количеством позиций (байтами) и к набору данных можно обращаться как через операторы форматного ввода-вывода, так и через операторы бесформатного ввода-вывода. Символ E означает, что физические записи измеряются количеством позиций (байтами) и к набору данных можно обращаться только через операторы форматного ввода-вывода.

Наборы данных прямого доступа, предназначенные для хранения данных в символьных формах, открываются при первом обращении к ним через оператор ввода или оператор вывода, причем они открываются как обновляемые. Закрытие этих наборов осуществляется без специальных указаний после окончания выполнения программы.

8.1.1. Оператор форматного вывода прямого доступа. Оператор форматного вывода прямого доступа имеет вид:

$$n\ \text{WRITE}(a'r,b)c$$

где n — либо пусто, либо метка оператора вывода; a — ссылочный номер набора данных; r — порядковый номер физической записи; b — метка оператора формата или имя массива формата; c — список выводимых величин.

Оператор вывода осуществляет перевод в символьную форму и засылку значений величин, указанных в списке c, в физические записи r, $r+1$ и т. д. таким образом, что в каждую физическую запись помещаются значения величин, относящиеся к одной логической записи. Напомним, что указания о делении величин, входящих в список c, на логические записи, содержатся в операторе формата b (в массиве формата b).

Перевод значений величин осуществляется с учетом форматов этих величин, содержащихся в операторе формата с меткой b (в массиве формата с именем b), а также с учетом типов и длин этих величин.

После выполнения оператора вывода переменная связи, указанная в операторе описания набора данных, примет значение,

равное $r + t$, где t — количество физических записей, в которые осуществлялась засылка значений величин.

Рассмотрим частные случаи, возникающие в процессе выполнения оператора форматного вывода.

Случай 1. Длина логической записи равна длине физической записи. Тогда все позиции физической записи будут заполнены значениями величин, образующих логическую запись.

Случай 2. Длина логической записи меньше длины физической записи. Тогда значения величин, образующих логическую запись, размещаются в начале физической записи (в первых ее позициях), а ее остаток будет заполнен пробелами (символами «пробел»).

Случай 3. Список выводимых величин отсутствует. Тогда в запись r помещаются либо пробелы, либо последовательности символов, входящие в состав символьных форматов.

Случай 4. В операторе формата встретилась косая черта, означающая пропуск логической записи. Тогда физическая запись заполняется пробелами.

Не допускается, чтобы длина логической записи превосходила длину физической записи.

Рассмотрим несколько примеров.

Пример 1. Пусть задана программа:

```
      INTEGER  X(40)/40*5/,Y(20)/20*7/,K
      DEFINE FILE 9(4,480,E,K)
    1 FORMAT (40I10,20I4)
      WRITE (9'2,1)X,Y
      RETURN
      END
```

В результате выполнения этой программы будет образована одна логическая запись, в которую войдут значения всех элементов массивов X и Y. Эти значения займут в ней 480 позиций (400 позиций займут значения элементов массива X и 80 позиций — значения элементов массива Y), что не превышает длины физической записи. Следовательно, все эти данные будут помещены в одну физическую запись с порядковым номером 2. Переменная связи K примет значение, равное 3.

Пример 2. Пусть задана программа:

```
      INTEGER  X(40)/40*5/,Y(20)/20*7/,K
      DEFINE FILE  9(4,480,E,K)
    1 FORMAT  (35I10)
```

Оператор описания наборов данных и оператор поиска записи выполняют те же функции и имеют тот же вид, что и соответствующие операторы бесформатного ввода-вывода. Исключением является лишь то обстоятельство, что в операторе описания наборов данных, относящемся к группе операторов форматного ввода, в качестве указателя единицы измерения длины физической записи используются символы L и E.

Символ L означает, что физические записи измеряются количеством позиций (байтами) и к набору данных можно обращаться как через операторы форматного ввода-вывода, так и через операторы бесформатного ввода-вывода. Символ E означает, что физические записи измеряются количеством позиций (байтами) и к набору данных можно обращаться только через операторы форматного ввода-вывода.

Наборы данных прямого доступа, предназначенные для хранения данных в символьных формах, открываются при первом обращении к ним через оператор ввода или оператор вывода, причем они открываются как обновляемые. Закрытие этих наборов осуществляется без специальных указаний после окончания выполнения программы.

8.1.1. Оператор форматного вывода прямого доступа. Оператор форматного вывода прямого доступа имеет вид:

n WRITE(*a'r*,*b*)*c*

где *n* — либо пусто, либо метка оператора вывода; *a* — ссылочный номер набора данных; *r* — порядковый номер физической записи; *b* — метка оператора формата или имя массива формата; *c* — список выводимых величин.

Оператор вывода осуществляет перевод в символьную форму и засылку значений величин, указанных в списке *c*, в физические записи *r*, *r* + 1 и т. д. таким образом, что в каждую физическую запись помещаются значения величин, относящиеся к одной логической записи. Напомним, что указания о делении величин, входящих в список *c*, на логические записи, содержатся в операторе формата *b* (в массиве формата *b*).

Перевод значений величин осуществляется с учетом форматов этих величин, содержащихся в операторе формата с меткой *b* (в массиве формата с именем *b*), а также с учетом типов и длин этих величин.

После выполнения оператора вывода переменная связи, указанная в операторе описания набора данных, примет значение,

равное $r + t$, где t — количество физических записей, в которые осуществлялась засылка значений величин.

Рассмотрим частные случаи, возникающие в процессе выполнения оператора форматного вывода.

Случай 1. Длина логической записи равна длине физической записи. Тогда все позиции физической записи будут заполнены значениями величин, образующих логическую запись.

Случай 2. Длина логической записи меньше длины физической записи. Тогда значения величин, образующих логическую запись, размещаются в начале физической записи (в первых ее позициях), а ее остаток будет заполнен пробелами (символами «пробел»).

Случай 3. Список выводимых величин отсутствует. Тогда в запись r помещаются либо пробелы, либо последовательности символов, входящие в состав символьных форматов.

Случай 4. В операторе формата встретилась косая черта, означающая пропуск логической записи. Тогда физическая запись заполняется пробелами.

Не допускается, чтобы длина логической записи превосходила длину физической записи.

Рассмотрим несколько примеров.

Пример 1. Пусть задана программа:

```
      INTEGER  X(40)/40*5/,Y(20)/20*7/,K
      DEFINE FILE 9(4,480,E,K)
    1 FORMAT (40I10,2014)
      WRITE (9'2,1)X,Y
      RETURN
      END
```

В реаультате выполнения этой программы будет образована одна логическая запись, в которую войдут значения всех элементов массивов X и Y. Эти значения займут в ней 480 позиций (400 позиций займут значения элементов массива X и 80 позиций — значения элементов массива Y), что не превышает длины физической записи. Следовательно, все эти данные будут помещены в одну физическую запись с порядковым номером 2. Переменная связи K примет значение, равное 3.

Пример 2. Пусть задана программа:

```
      INTEGER  X(40)/40*5/,Y(20)/20*7/,K
      DEFINE FILE  9(4,480,E,K)
    1 FORMAT  (35I10)
```

```
DEFINE FILE 5(30,15,E,K)
FORMAT ('ПЕТРОВ',7H,ИВАНОВ)
WRITE (5'7,3)
RETURN
END
```

7. Определить последовательности символов и номера физических записей, в которые будут помещены эти последовательности в результате выполнения следующей программы:

```
LOGICAL C
DEFINE FILE 8(50, 15, E,L)
FORMAT (2F5.2/2I4/L2)
A=—2.3
B=17.08
K=123
M=—35
C=.FALSE.
WRITE (8'3,3) A, B, K,M,C
STOP
END
```

8. Пусть четвертая физическая запись набора данных прямого доступа с номером 8 содержит строку символов

01234567—24.37—47.625800__________________

а пятая — строку символов

—_1_27_A389389XYZ________________________

Указать, какие символы будут содержать эти записи после выполнения следующей программы:

```
  DEFINE FILE 8(40,40,E,L)
4 FORMAT (T6,'11_11_11_'/3X,I5)
  K=—135
  WRITE (8'4,4)K
  RETURN
  END
```

9. Пусть заданы два набора данных прямого доступа. Первый набор имеет ссылочный номер 1 и содержит 15 физических записей длиной по 12 позиций. Первая физическая запись этого набора содержит значения переменных A,B и C, которые представлены в машинной форме. Переменные относятся к целому типу и имеют стандартную длину.

Второй набор имеет ссылочный номер 2 и содержит четыре физические записи длиной по пятьдесят позиций.

а) Составить программу, осуществляющую перепись значений переменных A,B и C из первого набора во второй и удовлетворяющую следующим условиям: значения переменных A,B,C должны помещаться во вторую физическую запись второго набора в символьном виде (в формате I5); эти значения должны размещаться в записи, начиная с первой позиции; оставшиеся позиции записи должны заполняться пробелами.

б) Составить программу, осуществляющую перепись значений переменных A,B и C из первого набора во второй и удовлетворяющую следующим условиям: значения переменных A,B и C должны помещаться во вторую физическую запись второго набора данных в символьном виде (в формате I5); эти значения должны помещаться в запись, начиная с одиннадцатой позиции; первые десять позиций записи и оставшиеся позиции записи должны заполняться символами * (звездочка).

10. Составить программу, обеспечивающую умножение вещественных векторов $a(a_1, a_2, \ldots, a_{10})$ и $b(b_1, b_2, \ldots, b_{10})$ по формуле $c_i = a_i \times b_i (i = 1,2, \ldots, 10)$ и вывод результатов в первую физическую запись некоторого набора данных прямого доступа в следующем виде:

$$C = c_1, c_2, \ldots, c_{10}$$

где C — символ (буква), который используется для обозначения результирующего вектора; $c_i (i = 1,2, \ldots, 10)$ — значение i-й компоненты вектора C, представленное в формате F7.5.

Для определенности считать, что

$a_1 = a_2 = \ldots = a_{10} = 3.5;$
$b_1 = b_2 = \ldots = b_{10} = 7.8.$

8.1.2. Оператор форматного ввода прямого доступа. Оператор форматного ввода прямого доступа имеет вид:

$$n \text{ READ } (a'r,b)c$$

где n — либо пусто, либо метка оператора ввода; a — ссылочный номер набора данных, r — порядковый номер физической записи; b — либо метка оператора формата, либо имя массива формата; c — список вводимых величин.

Оператор ввода осуществляет выборку значений величин из физических записей r, $r + 1$ и т. д., их перевод в машинную форму и присваивание элементам списка вводимых величин. Переход от одной физической записи к другой осуществляется после того, как

из очередной физической записи будут выбраны все данные, относящиеся к одной логической записи. Это определяется по форматному выражению, входящему в состав оператора формата *b* (массива формата *b*). Процесс перевода значений величин осуществляется с учетом форматов этих величин, содержащихся в операторе формата с меткой *b* (в массиве формата с именем *b*), и с учетом их типов и длин.

После выполнения оператора ввода переменная связи, указанная в операторе описания набора данных, примет значение равное $r+t$, где t — количество физических записей, из которых выбраны данные.

Список *c* может отсутствовать. В этом случае запись либо пропускается, либо из нее выбираются символы, которые помещаются в форматное выражение (на место символов, содержащихся в символьных форматах). Это зависит от содержимого форматного выражения.

Рассмотрим несколько примеров.

Пример 1. Пусть задана программа:

```
      DEFINE FILE 7(20,70,L,J)
    9 FORMAT (6F8.2)
      READ (7'10,9)A,B,C,R,S,Q
      RETURN
      END
```

и пусть десятая запись набора данных с номером 7 содержит символы:

␣␣␣ 64.30␣␣−18.07␣␣␣␣9.99␣␣␣−6.40␣␣␣24.87
␣␣␣−0.10␣␣␣␣␣.00␣␣␣86.40␣␣␣␣␣␣

Тогда в результате выполнения этой программы переменные A,B,C,R,S,Q получат следующие значения:

A=64.30,B=−18.07, C=9.99, R=−6.40, S=24.87, Q=−0.10

Пример 2. Пусть заданы:

а) программа

```
      REAL A(10)/−14.36,208.16,−307.01,0.00,3*−16.01,
           2*5.23,0.34/
      DEFINE FILE 5(35,45,E,J)
    20 FORMAT (5F7.2)
      READ (5'6,20)(A(I), I=2,10,2)
      READ (5'7,20)(A(I),I = 1,9,2)
      RETURN
      END
```

б) последовательность символов

__ —6.30__ __ __3.01__ —15.00__116.092104.00—776.08__ __ __

содержащаяся в шестой физической записи набора данных с номером 5;

в) последовательность символов

8001.36—816.04—209.001000.109863.99—733.00__ __ __

содержащаяся в седьмой физической записи набора данных с номером 5.

Тогда в результате выполнения приведенной выше программы элементы массива A примут следующие значения:

A(1)=8001.36, A(2)=—6.30, A(3)=—816.04, A(4)=3.01, A(5)=—209.0, A(6)=—15.0, A(7)=1000.1, A(8)=116.09, A(9)=9863.99, A(10)=2104.0

Пример 3. Пусть задана программа:

```
   DEFINE FILE 6(10,30,L,I)
12 FORMAT ('CHARLAMOV')
   READ (6'4,12)
   RETURN
   END
```

и пусть четвертая запись набора данных с номером 6 содержит последовательность символов

RETROVFORAISTABCDEFKLMN__ __ __ __ __ __ __

В результате выполнения приведенной выше программы оператор формата с меткой 12 примет вид:

```
12 FORMAT ('PETROVFOR')
```

Вопросы и упражнения

1. Чем отличаются (по выполняемым функциям) операторы форматного ввода прямого доступа от оператора бесформатного ввода?

2. Может ли оператор форматного ввода использоваться для выборки из физической записи только части содержащихся в ней данных?

3. Пусть задана программа:

```
   LOGICAL X
   DEFINE FILE 4 (4,20,E,L)
1  FORMAT (3G4.2)
```

```
      READ (4'3,1)I,X
      STOP
      END
```

и пусть третья запись набора данных с номером 4 содержит символы

—140_ _ _ Т3.1401234879

Определить значения переменных I,X, которые они получат в результате выполнения данной программы.

4. Пусть заданы два набора данных прямого доступа. Первый набор имеет ссылочный номер 1 и содержит двадцать физических записей длиной по 48 позиций, в которых хранятся по восемь целых констант в формате I6.

Второй набор данных имеет ссылочный номер 2 и содержит десять физических записей по тридцать две позиции.

Составить программу переписи значений величин из 1-го набора во второй таким образом, чтобы в первую запись 2-го набора попали значения шести первых величин из первой и значения двух первых величин из второй записей 1-го набора; во вторую запись 2-го набора попали значения шести первых величин из третьей и двух первых величин из четвертой записей 1-го набора и т.д. Значения величин во втором наборе должны быть представлены в формате I4. Программу составить в двух вариантах: с использованием форматов типа I и без использования форматов типа I.

5. Пусть набор данных прямого доступа со ссылочным номером 8 имеет десять записей с длиной, равной 60 байтам. Восьмая запись этого набора данных имеет вид:

_ _ _ _ _ _ —246_0.362E_04_0.623_ _ _ _ _ _5.27
_ _ _ _ _0.222E—01_ _24.5_ _ _ _

Определить значения переменных I,X,A,B,C,R, которые они получат в результате выполнения следующей программы:

```
      DEFINE FILE 8(10, 60, E, L)
    2 FORMAT  (6G10.3)
      READ (8'8,2)I,X,A,B,C,R
      RETURN
      END
```

6. Пусть набор данных прямого доступа со ссылочным номером 20 имеет 15 записей с длиной, равной 50 байтам. Вторая запись этого набора содержит целые числа 312, —17, 94, 20, —4,—862, 46,

207, —22, 24 и имеет вид:

__312__—17____94____20____—4__862____46__
207__—22____24

Десятая запись содержит числа 102, —84, —0.00268, 75200.0, 2.56, —37.8 и имеет вид:

__102__—84—0.268E—02_0.752E _ 05 _ 0.256E _ 01
—0.378E_02

Одиннадцатая запись содержит числа 200.0, —3.52, 8.25, —0,0467, 3.42 и имеет вид:

_0.200E_03—0.352E_01_0.825E_01—0.467E—01_0.342
E_01

Составить программу, которая осуществляет ввод приведенных выше чисел и присваивает их элементам одномерного массива M целого типа стандартной длины и элементам одномерного массива X вещественного типа стандартной длины следующим образом:

M(1) = 102, M(2) = —84, M(3) = 312, M(4) = —17, M(5) = 94, M(6) = 20, M(7) = —4, M(8) = 862, M(9) = 46, M(10) = 207, M(11)—22, M(12) = 24, X(1) = —0.00268, X(2) = 75200.0, X(3) = 2.56, X(4) = = —37.8, X(5) = 200.0, X(6) = —3.52, X(7) = 8.25, X(8) = —0.0467

8.2. Операторы форматного ввода-вывода последовательного доступа

Операторы форматного ввода-вывода последовательного доступа предназначены для обмена данными между наборами с последовательным доступом и оперативной памятью. В фортран-программах могут использоваться наборы данных с последовательным доступом, имеющие следующие форматы блоков: F, FB, V, VB, U.

К группе операторов форматного ввода-вывода последовательного доступа относятся следующие операторы: *оператор форматного вывода последовательного доступа*, *оператор форматного ввода последовательного доступа*, *оператор возврата на одну логическую запись*, *оператор возврата к первой логической записи набора*.

Оператор возврата на одну логическую запись и оператор возврата к первой логической записи набора выполняют те же функции и имеют тот же вид, что и соответствующие операторы бесформатного ввода-вывода.

Наборы данных последовательного доступа открываются при первом обращении к ним через оператор ввода или оператор вы-

вода. При выполнении оператора ввода набор данных последовательного доступа открывается как входной, а при выполнении оператора вывода — как выходной. При выполнении оператора возврата к первой логической записи набора (REWIND) осуществляется закрытие набора данных. Закрытие набора данных последовательного доступа осуществляется также после окончания выполнения программы.

8.2.1. Оператор форматного вывода последовательного доступа. Этот оператор может быть представлен в одной из следующих форм:

n WRITE (*a*,*b*)*c*
n WRITE (*a*,*b*)

Здесь *n* — либо пусто, либо метка помечающая данный оператор; *b* — метка оператора формата или имя массива формата; *c* — список выводимых величин; *a* — ссылочный номер набора данных.

Примеры операторов вывода:

```
15 WRITE (8,5)A,B,C
   WRITE (M,1) X,Y(5),(Z(K),K=I,J)
```

Оператор вывода в символьной форме выполняет те же функции, что и оператор вывода в машинной форме. Однако он имеет следующие особенности:

1) Преобразует значения величин, перечисленных в списке *c*, в символьный вид. Символьные формы величин описываются в операторе формата *b* (в массиве формата *b*).

2) В процессе его выполнения могут создаваться не одна, а несколько последовательных логических записей. Это также указывается в операторе формата *b* (в массиве формата *b*). Длина этих записей не должна превышать установленной для данного набора максимальной длины физической записи.

3) Отсутствие списка выводимых величин не всегда приводит к созданию фиктивной записи. Это определяется оператором формата (массивом формата).

4) Если набор данных был открыт оператором форматного вывода, то в дальнейшем (до его закрытия) данные в этот набор должны засылаться только оператором форматного вывода.

Рассмотрим несколько ситуаций, возникающих в процессе выполнения форматного оператора вывода последовательного доступа.

С л у ч а й 1. а) Набор данных имеет формат блоков F. Длина логической записи равна длине физической записи. Тогда все поз-

ции физической записи будут заполнены данными, составляющими логическую запись. Данные будут преобразованы в соответствии с оператором формата (массива формата).

б) Набор данных имеет формат блоков V, максимальную длину записи, равную l байтам, максимальную длину блока, равную $l+4$ байтам. Длина логической записи равна $l-4$ байтам. Тогда физическая запись будет иметь максимальную длину, равную l байтам. Первые четыре байта будут содержать служебную информацию, а остальные $l-4$ байта будут заполнены данными, составляющими логическую запись. Длина блока будет равна $l+4$ байтам.

в) Набор данных имеет формат блоков U и максимальную длину блоков, равную l байтам. Длина логической записи равна l байтам. Тогда все позиции физической записи будут заполнены данными, составляющими логическую запись. Длина блока будет равна длине физической записи.

Случай 2. а) Набор данных имеет формат блоков F. Длина логической записи меньше длины физической записи. Тогда данные, составляющие логическую запись, размещаются в начале физической записи (в первых ее позициях), а ее остаток будет заполнен символами «пробел».

б) Набор данных имеет формат блоков V, максимальную длину записи, равную l байтам, и максимальную длину блока, равную $l+4$ байтам. Длина логической записи равна l_1 байтам, и $l_1 < l-4$. Тогда длина физической записи будет равна l_1+4 байтам. Длина блока будет равна l_1+8 байтам.

в) Набор данных имеет формат блоков U и максимальную длину блока, равную l байтам. Длина логической записи равна l_1 байтам, и $l_1 < l$. Тогда длина физической записи будет равна длине логической записи, т. е. l_1 байтам. Длина блока также будет равна l_1 байтам.

Случай 3. В форматном операторе вывода последовательного доступа отсутствует список выводимых величин.

а) Набор данных имеет формат блоков F. В физическую запись помещаются либо символы «пробел», либо последовательности символов, входящие в состав символьных форматов. Тогда длина блока будет равна длине физической записи.

б) Набор данных имеет формат блоков V, максимальную длину записи, равную l байтам, максимальную длину блока, равную $l+4$ байтам. В этом случае либо создается фиктивная физическая запись, не содержащая данных, либо создается запись, содержащая последовательность символов, указанных в символьных форматах. При вводе данных фиктивные записи должны пропускаться (либо

с помощью выполнения оператора ввода без списка вводимых величин, либо с помощью указания пропуска записи в операторе формата). Длина физической записи будет равна $l_1 + 4$ байтам, где l_1 при отсутствии символьных форматов будет равна нулю. Длина блока соответственно будет равна $l_1 + 8$ байтам.

в) Набор данных имеет формат блоков U, максимальную длину блоков, равную l байтам. В этом случае либо создается фиктивная физическая запись, не содержащая данных, либо в записи размещаются последовательности символов, содержащиеся в символьных форматах. Фиктивные записи при вводе данных должны пропускаться.

Случай 4. В операторе формата (массиве формата) указан пропуск логической записи (символ / — косая черта).

а) Набор данных имеет формат блоков F. Тогда физическая запись заполняется символами «пробел».

б) Набор данных имеет формат блоков V. Тогда создается фиктивная физическая запись, содержащая только служебную информацию.

в) Набор данных имеет формат блоков U. Тогда создается фиктивная физическая запись, не содержащая данных.

Случай 5. Набор данных имеет формат блоков FB. Тогда заполнение физических записей будет осуществляться таким же образом, как и при использовании формата F. Физические записи будут объединяться в блоки по n записей, где n — целое число, равное частному от деления длины блока на длину физической записи. Последний блок набора может содержать меньшее количество физических записей.

Случай 6. Набор данных имеет формат блоков VB. Тогда заполнение физических записей будет осуществляться таким же образом, что и при использовании формата V. Объединение физических записей в блоки происходит путем их накапливания в буфере, длина которого берется равной максимальной длине блока. Процесс накапливания прекращается, когда длина незаполненной части буфера становится меньше максимальной длины физической записи. Накопленные в буфере физические записи оформляются в виде блока и переписываются в набор данных, после чего в буфере будут накапливаться физические записи для следующего блока.

Рассмотрим несколько примеров.

Пример 1. Пусть заданы:

а) набор данных последовательного доступа со ссылочным номером 3, состоящий из блоков фиксированной длины (формат F), имеющих длину 400 байтов;

б) программа

```
      INTEGER X(40)/40*1/, Y(20)/20*2/,Z(40)/40*3/
    1 FORMAT (40I10)
      WRITE (3,1)X
      WRITE (3,1)Z,Y
      RETURN
      END
```

В результате выполнения этой программы в наборе данных с номером 3 будут образованы три физические записи. В первую физическую запись будут помещены значения элементов массива X, которые займут в ней все 400 байтов. Во вторую физическую запись будут помещены значения элементов массива Z, которые займут в ней также все 400 байтов. В третью физическую запись будут помещены значения элементов массива Y, которые займут в ней 200 байтов. Остальные 200 байтов будут заполнены пробелами. Каждая из созданных физических записей войдет в состав отдельного блока с длиной 400 байтов.

Пример 2. Пусть заданы:

а) набор данных последовательного доступа со ссылочным номером 3, состоящий из блоков переменной длины (формат V), имеющих максимальную длину 408 байтов;

б) программа, которая приведена в примере 1.

В результате выполнения этой программы в наборе данных будет создано 3 физических записи, каждая из которых войдет в состав отдельного блока. В первую физическую запись будут помещены значения элементов массива X, которые займут в ней 400 байтов, причем длина этой записи будет равна 404 байтам, а длина блока, в состав которого войдет запись — 408 байтам. Во вторую физическую запись будут помещены значения элементов массива Z, которые займут в ней также 400 байтов. Длина этой записи будет равна 404 байтам, а длина блока, в состав которого войдет запись — 408 байтам. В третью физическую запись будут помещены значения элементов массива Y, которые займут в ней 200 байтов. Длина этой записи будет равна 204 байтам, а длина блока, в состав которого войдет запись — 208 байтам.

Пример 3. Пусть заданы:

а) набор данных последовательного доступа со ссылочным номером 3, состоящий из блоков неопределенной длины (формат U), имеющих максимальную длину 400 байтов;

б) программа, которая приведена в примере 1.

В результате выполнения этой программы в наборе данных будет создано 3 физических записи, каждая из которых войдет в

состав отдельного блока. В первую запись будут помещены значения элементов массива X, которые займут в ней 400 байтов. Длина этой записи будет равна 400 байтам, длина блока, в состав которого войдет запись — 400 байтам. Во вторую запись будут помещены значения элементов массива Z, которые займут в ней также 400 байтов. Длина этой записи и длина блока, в состав которого войдет эта запись, будут равны 400 байтам. В третью запись будут помещены значения элементов массива Y, которые займут в ней 200 байтов. Длина этой записи и длина блока, в состав которой войдет эта запись, будут равны 200 байтам.

Пример 4. Пусть заданы:

а) набор данных последовательного доступа со ссылочным номером 1, состоящий из блоков фиксированной длины (формат F), имеющих длину 250 байтов;

б) программа

```
      REAL X(5,10)/50*0.5/,Y(3,10)/30*0.01/
    1 FORMAT (10X,'ЗНАЧЕНИЯ X,Y')
    2 FORMAT (//50F5.1//30F5.2)
    3 FORMAT (50F5.1)
      WRITE (1,1)
      WRITE (1,3)
      WRITE (1,2)X,Y
      RETURN
      END
```

В результате выполнения этой программы в наборе с номером 1 будет создано 7 физических записей. В первую из них будут помещены символы

_ _ _ _ _ _ _ _ _ _ _ ЗНАЧЕНИЯ_X,Y и 228 символов «пробел». Во вторую, третью, четвертую и шестую записи будет помещено по 250 пробелов, в пятую — значения элементов массива X, которые займут в ней все 250 байтов. В седьмую запись будут помещены значения элементов массива Y, которые займут в ней 150 байтов, и 100 символов «пробел». Каждая из созданных физических записей войдет в состав отдельного блока с длиной 250 байтов.

Пример 5. Пусть заданы:

а) набор данных последовательного доступа со ссылочным номером 1, состоящий из блоков переменной длины (формат V), имеющих максимальную длину 258 байтов;

б) программа, которая приведена в примере 4. В результате выполнения этой программы в наборе с номером 1 будет создано 7

физических записей, каждая из которых войдет в состав отдельного блока. В первую запись будут помещены десять символов «пробел» и символы ЗНАЧЕНИЯ_X,Y. Длина этой записи будет равна 26 байтам, а длина блока, в состав которого войдет запись — 30 байтам. Вторая, третья, четвертая и шестая записи будут созданы как фиктивные. Длина блоков, в состав которых войдут эти записи, будет равна 8 байтам. В пятую запись будут помещены значения элементов массива X, которые займут в ней 250 байтов. Длина этой записи будет равна 254 байтам, а длина блока, в состав которого войдет запись — 258 байтам. В седьмую запись будут помещены значения элементов массива Y, которые займут в ней 150 байтов. Длина этой записи будет равна 154 байтам, а длина блока, в состав которого войдет эта запись — 158 байтам.

Пример 6. Пусть заданы:

а) набор данных последовательного доступа со ссылочным номером 2. Формат блока — FB, длина физических записей — 60 байтов, длина блоков — 180 байтов;

б) программа

```
      REAL X(5,2)/10*0.1/,Y(5,2)/10*—1000.5),
     *     A(10)/10*0.5/,B(5)/5*—0.01/
      INTEGER M(3)/1,2,3/,N(5)/1,2,3,4,5/
    1 FORMAT (10(F5.1,1X)/16,2(F5.2,F10.2,2(I5,1X)))
    2 FORMAT (I6,2(F5.1,F10.2,2(I5,1X)))
    3 FORMAT (5F5.2)
    4 WRITE (2,1)A,M(1),((X(I,J),Y(I,J),N(I),N(J),
     *          J=1,2),I=1,2)
    5 WRITE (2,2)M(2),((X(I,J),Y(I,J),N(I),N(J),
     *          J=1,2),I=3,4)
    6 WRITE (2,2)M(3),(X(5,J),Y(5,J),N(5),N(J),J=1,2)
    7 WRITE (2,3)B
      RETURN
      END
```

В результате выполнения этой программы в наборе данных со ссылочным номером 2 будут созданы 7 физических записей, которые войдут в состав 3-х блоков. В первую запись будут помещены следующие данные: значение величины A(1), символ «пробел», значение величины A(2), символ «пробел» и т. д. (всего 10 таких пар). Эти данные займут в записи 60 байтов.

Во вторую запись будут помещены значения величин

M(1),X(1,1),Y(1,1),N(1),N(1),X(1,2),Y(1,2),N(1),N(2),

которым соответствует форматное выражение

16,2(F5.2,F10.2,2(I.5,1X)).

Эти значения займут в пей все 60 байтов.

В третью запись будут помещены значения величин

X(2,1),Y(2,1),N(2),N(1),X(2,2),Y(2,2),N(2),N(2),

которым соответствует форматное выражение

2(F5.2,F10.2,2(I5,1X)).

Эти значения займут в ней 54 байта.
Остальные 6 байтов будут заняты символом «пробел».

В четвертую запись будут помещены значения величин

M(2),X(3,1),Y(3,1),N(3),N(1),X(3,2),Y(3,2),N(3),N(2),

которым соответствует форматное выражение

16.2(F5.1,F10.2,2(I5,1X)).

Эти значения займут в пей 60 байтов.

В пятую запись будут помещены значения величин

X(4,1),Y(4,1),N(4),N(1),X(4,2),Y(4,2),N(4),N(2),

которым соответствует форматное выражение

2(F5.2,F10.2,2(I5,1X)).

Эти значения займут в ней 54 байта. Остальные 6 байтов будут заняты символом «пробел».

В шестую запись будут помещены значения величин

M(3),X(5,1),Y(5,1),N(5),N(1),X(5,2),Y(5,2),N(5),N(2),

которым соответствует форматное выражение

16,2(F5.1,F10.2,2(I5,X)).

Эти значения займут в ней все 60 байтов.

В седьмую запись будут помещены значения элементов массива B, которым соответствует форматное выражение

5F5.2.

Эти значения займут в ней двадцать пять байтов. Остальные 35 байтов будут заполнены символом «пробел». Записи 1, 2 и 3 войдут в состав 1-го блока, записи 4, 5 и 6 — в состав 2-го блока. Третий

(последний) блок будет содержать всего лишь одну (седьмую) физическую запись.

Пример 7. Пусть заданы:

а) набор данных последовательного доступа со ссылочным номером 2. Формат блоков набора — VB, максимальная длина блоков — 226 байтов, максимальная длина физической записи — 74 байта;

б) программа, которая приведена в примере 6.

В результате выполнения этой программы в наборе данных со ссылочным номером 2 будут созданы 7 физических записей. Записи 1, 2 и 3 будут иметь длины 64, 64 и 58 байтов соответственно. Эти записи войдут в состав первого блока, длина которого будет равна 190 байтам. Записи 4, 5 и 6 будут иметь длины 64, 58 и 64 байта соответственно. Эти записи войдут в состав второго блока, длина которого будет равна 190 байтам. Запись 7 будет иметь длину 29 байтов, которая войдет в состав третьего блока. Длина этого блока будет равна 33 байтам.

Вопросы и упражнения

1. Какой формат блоков может иметь последовательный набор данных, если данные помещаются в этот набор с помощью операторов форматного вывода?

2. Пусть имеется программа:

```
LOGICAL L1,L2
  REAL A/—0.34/,B/4.03/,C(5)/5* —10.05/
1 FORMAT (2L2,7(1X,F6.2))
  L1=A.GT.B
  L2=B.GE.C(1)
  WRITE (7,1)L1,L2,A,B,C
  RETURN
  END
```

Написать последовательность символов, которая будет помещена в первую логическую запись седьмого набора данных.

3. Пусть заданы:

а) набор данных последовательного доступа со ссылочным номером 8; формат блоков набора — FB, длина блоков — 112 байтов, длина физических записей — 56 байтов;

б) программа

```
  REAL A(5,10)/50*0.05/
1 FORMAT('_МАТРИЦА_A')
2 FORMAT (10F5.2)
```

```
      DO 3 I=1,5
      A(1,1)=A(1,1)+A(I,1)
      DO 3 J=1,10
    3 A(I,J)=A(I,J) + A(1,1)
      WRITE (8,1)
      WRITE (8,2)
      WRITE (8,2)((A(I,J),J = 1,10),I=1,5)
      RETURN
      END
```

Ответить на следующие вопросы:

1) Сколько блоков будет создано в наборе с номером 8 в результате выполнения этой программы?

2) Сколько физических записей будет содержаться в каждом блоке набора?

3) Какие последовательности символов будут помещены в физические записи набора?

4. Пусть заданы:

а) набор данных последовательного доступа со ссылочным номером 8, формат блоков набора — VB, максимальная длина блоков — 140 байтов, максимальная длина физических записей — 68 байтов;

б) программа, которая приведена в примере 3.

Ответить на следующие вопросы:

1) Сколько блоков будет создано в наборе с номером 8 в результате выполнения этой программы?

2) Сколько физических записей будет содержаться в каждом блоке набора?

5. Пусть заданы:

а) одномерный массив M целых чисел стандартной длины, элементы которого

M(1)=—104, M(2)=91, M(3)=—36, M(4)=27, M(5)=1, M(6)=6, M(7)=87, M(8)=95, M(9)=—3, M(10)=18;

б) комплексные переменные стандартной длины

A = (36.5,—24.3) и B=(7.42, —0.17).

Составить программу записи значений элементов массива и переменных A и B в последовательный набор данных со ссылочным номером 9, удовлетворяющую следующим условиям:

1) в первую логическую запись набора должны попасть значения величин A,B,M(1),M(2),M(3),M(4),M(5);

2) во вторую логическую запись набора должны попасть значения величин M(6),M(7),M(8),M(9),M(10);

3) значения компонент переменных A и B должны представляться в формате E10.3, а значения элементов массива M — в формате I5.

6. Пусть заданы:

а) набор данных прямого доступа со ссылочным номером 1, состоящий из 10 физических записей длиной по 120 байтов. В каждой из этих записей содержится по 10 значений величин вещественного типа, которые представлены в формате D12.3 и размещаются в записях, начиная с первого байта;

б) набор данных последовательного доступа со ссылочным номером 2. Формат блоков набора F, длина блоков — 180 байтов.

1) Составить программу, осуществляющую перепись значений величин из первого набора во второй и удовлетворяющую следующим условиям: значения величин, содержащиеся в i-й записи первого набора ($i = 1, 2, \ldots, 10$), переписываются в i-ю запись второго набора; во втором наборе данных значения величин должны быть представлены в формате F15.3.

2) Составить программу, осуществляющую перепись значений величин из первого набора во второй и удовлетворяющую следующим условиям: в первую запись набора 2 помещаются все значения величин из первой записи и пять значений величин из второй записи набора 1, во вторую запись набора 2 помещаются все значения величин из третьей записи и пять значений величин из четвертой записи и т. д.; в наборе 2 значения величин должны быть представлены в формате F12.3.

7. Пусть значения элементов целого одномерного массива A(100) содержатся в первой и четвертой записях набора данных прямого доступа, который имеет следующие характеристики: ссылочный номер набора — 8, длина физической записи — 300 байтов, количество физических записей — 8. Значения элементов массива представлены в формате I6 и размещаются в записях следующим образом: в первой записи содержатся значения A(1),A(2), . . ., A(50); в четвертой — значения A(51), A(52), . . ., A(100).

Составить программу пересылки значений элементов массива A в физические записи 5, 6, 7 и 8 последовательного набора данных таким образом, чтобы выполнялись следующие требования:

а) в запись 5 должны попасть значения A(1), A(2), . . ., A(25), в запись 6 — значения A(26), A(27), . . ., A(50) и т. д.;

б) значения элементов массива должны размещаться в записях в формате I5, начинаться с 11-й позиции и должны отделяться друг от друга пятью символами * (звездочка);

в) оставшиеся позиции записей должны заполняться пробелами.

Последовательный набор данных имеет следующие характеристики: ссылочный номер 9, формат блоков F, длину блоков 300 байтов.

8. Пусть задана программа:

```
      COMPLEX C/(3.5,—0.6)/
      LOGICAL L
      DIMENSION M(3),B1(3)
      REAL B(3)/—2.5,1.7,0.8/,R(3)/3*0.1/
    1 FORMAT (L3, 2(F5.1,2X),
     (E10.3,2(F5.1,2X),I4)
      DO 5 I=1,3
      M(I)=I
    5 B1(I)=B(I)/R(I)+1.5
      L=B1(1).LT.0.0
      WRITE (10,1)L,C,(B1(I),B(I),R(I),M(I),I=1,3)
      RETURN
      END
```

Определить:

а) Сколько логических записей будет создано в наборе данных с номером 10 в результате выполнения этой программы?

б) Какие символы будут помещены в эти записи?

8.2.2. Оператор форматного ввода последовательного доступа. Этот оператор может быть представлен в одной из следующих форм:

n READ (*a*,*b*,END=*m*,ERR=*k*)*c*
n READ (*a*,*b*,END=*m*,ERR=*k*)
n READ (*a*,*b*,END=*m*)*c*
n READ (*a*,*b*,END=*m*)
n READ (*a*,*b*,ERR=*k*)*c*
n READ (*a*,*b*,ERR=*k*)
n READ (*a*,*b*)*c*
n READ (*a*,*b*)

Здесь *n* — либо пусто, либо метка, помечающая данный оператор; *a* — ссылочный номер набора данных; *b* — либо метка оператора формата, либо имя массива формата; *m*, *k* — метки выполняемых операторов; *c* — список вводимых величин.

Примеры операторов ввода:

```
    3 READ (3,7,END=8)X,Y,Z
      READ (N,15)
```

Оператор ввода в символьной форме выполняет те же функции, что и оператор ввода в машинной форме. Однако он имеет следующие особенности:

1) Преобразует введенные данные из символьного вида в машинный. Символьные формы данных описываются в операторе формата *b* (массиве формата *b*).

2) В процессе его выполнения могут вводиться данные не из одной, а из нескольких последовательно расположенных логических записей. Это также указывается в операторе формата *b* (в массиве формата *b*).

3) Отсутствие списка вводимых величин не всегда приводит к пропуску очередной логической записи. Это определяется оператором формата (массивом формата).

Если форматное выражение содержит символьные форматы, то из физической записи выбираются символы и помещаются в форматное выражение.

Параметры END=*m* и ERR=*k* могут отсутствовать. Если параметр END=*m* указан в операторе ввода и при выполнении этого оператора будет обнаружен конец набора данных, то выполнение программы будет продолжено с оператора, имеющего метку *m*. При отсутствии этого параметра в аналогичной ситуации выполнение программы прекращается. Если параметр ERR=*k* указан в операторе ввода и при выполнении этого оператора будет обнаружена ошибка, то выполнение программы будет продолжено с оператора, имеющего метку *k*. При отсутствии данного параметра в подобной ситуации выполнение программы будет прекращено. Рассмотрим несколько примеров.

Пример 1. Пусть первая физическая запись последовательного набора данных с номером 9 содержит символы

␣␣␣—0.8762E—03␣␣␣␣0.1200E—04␣␣␣␣0.9003E
␣02␣␣␣—0.9003E␣06␣␣␣␣0.9600E␣00␣␣␣—0.8000E
—02␣␣␣—99

Тогда в результате выполнения программы

```
6 FORMAT (6E14.4)
  READ (9,6)A,B,C,R,T,S
  RETURN
  END
```

переменные A,B,C,R,T,S получат следующие значения:

$A=-0.8762\cdot10^{-3}$, $B=0.12\cdot10^{-4}$, $C=0.9003\cdot10^{2}$, $R=-0.9003\cdot10^{6}$, $T=0.96$, $S=-0.8\cdot10^{-2}$.

Пример 2. Пусть заданы:
а) программа

```
      INTEGER *2 N2,N3,K2,K3
      DIMENSION X(8), Y(8)
    5 FORMAT (3X,F5.2,(I5,8E13.5),I2)
      READ (17,5) A,N2,X,N3,K2,Y,K3
      RETURN
      END
```

б) последовательный набор данных с номером 17, содержащий две физические записи. Первая физическая запись содержит символы

```
     1.05   57  0.24000E 00 -0.36210E 02
-0.88000E 01  0.20900E-0.1  0.64610E 03 -0.39470E
02 -0.10380E 03  0.99840E 0220
```

Вторая физическая запись содержит символы

```
   58  0.51064E 02  0.29300E 02  0.29300E
 02  0.29300E 02 -0.80300E 01 -0.84000E 00
0.76000E 00 -0.12801E 0321
```

В результате выполнения приведенной в пункте а) программы величины A,X,Y,N2,N3,K2,K3
получат следующие значения:

A=1.05, N2=57, X(1)=0.24, X(2)=—36.21, X(3)=—8.8, X(4)=0.0209, X(5)=646.1, X(6)=—39.47, X(7)=—103.8, X(8)=—99.84, N(3)=20, K2=58, Y(1)=51.064, Y(2)=29.3, Y(3)=29.3, Y(4)=29.3, Y(5)=—8.03, Y(6)=—0.84, Y(7)=0.76, Y(8)=—128,01, K3=21.

Пример 3. Пусть заданы:
а) программа

```
      DIMENSION M(5)
    6 FORMAT (5X,315/215)
    7 FORMAT ('МАССИВ   ', 3I5. 5I5)
      READ (10,7)
      READ (10,6)
      READ (10,6)M
      WRITE (11,7)M
      RETURN
      END
```

б) последовательный набор данных с номером 10, содержащий четыре физические записи. Первая физическая запись содержит последовательность символов

ТАБЛИЦА_N_1___50___51___52___53

Вторая физическая запись является фиктивной и не содержит данных. Третья физическая запись содержит символы

___197_1023_____3_____5_____2___

Четвертая физическая запись содержит символы

____15____99_____9_123710915___

в) последовательный набор данных с номером 11, который создается в процессе выполнения приведенной выше программы.

После выполнения оператора READ (10,7) оператор формата с меткой 7 будет иметь вид:

7 FORMAT ('ТАБЛИЦА_', 3HN_1,5I5)

Оператор READ (10,6) позволяет пропустить фиктивную физическую запись. В результате выполнения оператора READ (10,6)M элементы массива M примут значения:

M(1)=1023,M(2)=3,M(3)=5,M(4)=15,M(5)=99.

Оператор WRITE (11,7)M поместит в последовательный набор данных с номером 11 одну логическую запись, которая будет содержать последовательность символов

ТАБЛИЦА_N_1_1023_____3_____5____15____99

П р и м е р 4. Пусть заданы:

а) программа

```
5 FORMAT(8F10.4)
1 FORMAT(8E10.3)
  DIMENSION A(8)
2 READ(5,1,END=3)A
  WRITE(2,5)A
  GO TO 2
3 RETURN
  END
```

б) последовательный набор данных со ссылочным номером 5, который представляет собой последовательность данных, пробитых на перфокартах. Первая перфокарта содержит символы

_0.625E_01_0.996E_02_0.405E_03_0.640E_03_0.909E_02_0.309E—01_0.200E_03_0.699E_04

Вторая перфокарта содержит символы

_0.304E_02_0.399E—01_0.940E—01_0.664E_04_0.609E
_05_0.908E_03_0.600E_03_0.760E_00

Третья перфокарта содержит признак конца набора данных, т. е. последовательность символов /* в первой и второй позициях соответственно.

в) последовательный набор данных с номером 2, создаваемый при выполнении данной программы.

В результате выполнения программы в набор данных с номером 2 будут помещены две логические записи. Первая логическая запись будет содержать символы:

_ _ _ _6.2500_ _ _99.6000_ _405.0000_ _640.0000_ _
_90.9000_ _ _ _0.0309_ _200.0000_6990.0000

Вторая логическая запись будет содержать символы:

_ _ _30.4000_ _ _ _0.0399_ _ _ _0.0940_6640.000060900
0000_ _908.0000_ _600.0000_ _ _ _0.7600

Вопросы и упражнения

1. Пусть первая логическая запись 14-го набора данных содержит символы

_—1534_0.241E_01_ _—0.890E—01_ _ _ _ _394_ _
_T_ _69_ _58—119_ _—4

Определить значения переменных C,B и элементов массива M1, которые они получат в результате выполнения программы

```
      INTEGER  M1(8)/—16,99,100,1,3,—14,2,20/
      COMPLEX  C
      LOGICAL *1B
    2 FORMAT (6X,2(E10.3,2X),6X,L4,4I4)
      READ (14,2)C,B,(M1(I),I=2,8,2)
      RETURN
      END
```

2. Пусть последовательный набор данных со ссылочным номером 8 содержит четыре логических записи. Первая логическая запись содержит символы

32324—3.1464_5,93411_421_431_

Вторая логическая запись содержит символы:

512_522_532_

Третья логическая запись содержит символы:

613_623_633_

Четвертая логическая запись содержит символы:

714_724_734_

Написать, какие значения будут иметь элементы массивов MAT, K и B после выполнения программы

```
1 FORMAT (I3,2(I2,F5.2),3(I3,1X))
  INTEGER *2MAT(3,4),K(3)
  REAL B(2)
  READ (8,1)K(3),(K(I),B(I),I=1,2),MAT
  RETURN
  END
```

3. Пусть последовательный набор данных со ссылочным номером 15 содержит три логических записи. Первая логическая запись содержит символы:

_ _ _ _ _ —0.2654E_0311—27.512249.213—54.2

Вторая — символы:

_57.3—68.9—76.8

Третья — символы:

21—34.222—45.523_ _0.2

Написать, какие значения будут иметь переменная A1 и элементы массивов NA,A,B в результате выполнения программы

```
  REAL A(2,3), B(3)
  INTEGER NA(6)
2 FORMAT (5X, E11.4,(3(I2,F5.1)/3F5.1))
  READ(15,2)A1,(NA(K),A(1,K),K=1,3),B,(NA(I+3),
       A(2,I),I=1,3)
  RETURN
  END
```

4. Пусть задан набор данных последовательного доступа со ссылочным номером 12, состоящий из 2-х логических записей. Первая запись содержит числа N1, N2, . . ., N8, а вторая — числа

N9,N10, . . .,N14. Числа N1 и N2 представлены в форме $Ц_1Ц_2$, где $Ц_1$ и $Ц_2$ — десятичные цифры. Числа N3, N4, . . ., N14 представлены в форме $sЦ_1.Ц_2Ц_3$, где s — либо символ «пробел», либо — минус, $Ц_i$ ($i = 1, 2, 3$) — десятичные цифры. Числа в логических записях отделены друг от друга символом *(звездочка).

Составить программу, которая присваивала бы переменным K и N, а также элементам двумерных массивов A(2,3) и B(3,2) следующие значения:

K=N1, N=N2, B(1,1)=N3, A(1,1)=N4, B(2,1)=N5, A(1,2)=N6, B(3,1)=N7, A(1,3)=N8, B(1,2)=N9, A(2,1)=N10, B(2,2)=N11,A(2,2)=N12, B(3,2)=N13, A(2,3)=N14.

Переменные N и K относятся к целому типу и имеют стандартную длину. Массивы A и B относятся к вещественному типу и имеют стандартную длину.

5. Пусть задан последовательный набор данных со ссылочным номером 1, состоящий из n логических записей ($1 \leqslant n \leqslant 10$). Логические записи имеют одинаковую длину и каждая из них содержит пятьдесят вещественных значений, представленных в формате D14.5. Значения размещаются в записях, начиная с 5-й позиции.

Составить программу, которая вычисляет сумму значений, хранящихся в наборе и присваивает ее переменной A, имеющей вещественный тип и нестандартную длину.

8.2.3. Особенности управления выводом на устройство печати. Устройство печати относится к устройствам последовательного доступа. На нем можно создавать наборы данных только с последовательной организацией. В этих наборах под физической записью понимается одна печатная строка. При выводе данных на устройство печати в символьной форме (с помощью форматов) можно пользоваться двумя режимами его работы: без управляющих символов и с управляющими символами.

В режиме работы без управляющих символов все выводимые символы печатаются на бумажной ленте. Переход к следующей логической записи сопровождается переводом бумажной ленты на следующую строку.

В режиме работы с управляющими символами первый символ каждой записи не печатается. Он управляет движением бумаги по следующим правилам:

1. Первый символ — пробел. Бумага продвигается на одну строку.

2. Первый символ — цифра нуль. Бумага продвигается на две строки.

3. Первый символ — знак плюс. Бумага не продвигается.

Кроме того, в качестве управляющих символов могут использоваться цифры от 1 до 8. В этих случаях характер продвижения бумаги определяется размещением кодов этих цифр на управляющей ленте АЦПУ. Обычно цифра 1 используется для продвижения бумаги до первой строки на следующей странице.

Режимы работы устройств печати указываются в управляющих операторах задания.

При использовании формата типа Т следует помнить, что в режиме печати с управляющими символами отсчет позиций в логической записи начинается с управляющего символа. Рассмотрим несколько примеров.

Пример 1. Пусть задан фрагмент программы:

```
      DIMENSION X(5),Y(5)
      X(1)=0.5
      H=0.5
      Y(1)=X(1)**2+1
      DO 1 K=2,5
      X(K)=X(K-1)+H
    1 Y(K)=X(K)**2+1
    2 FORMAT ('1','_____ТАБЛИЦА_N1'/'0','_____X
     1      _______Y___'/('_,'2F7.2))
      WRITE (6,2)(X(I),Y(I),I=1,5)
```

Если устройство печати работает в режиме с управляющими символами, то в результате выполнения этого фрагмента программы на печать (в набор с номером 6) будут выданы строки символов, приведенные на рис. 8.1.

				Т	А	Б	Л	И	Ц	А		N	1	
				X							У			
			0	.	5	0				1	.	2	5	
			1	.	0	0				2	.	0	0	
			1	.	5	0				3	.	2	5	
			2	.	0	0				5	.	0	0	
			2	.	5	0				7	.	2	5	

Рис. 8. 1. Первый пример размещения данных на бумажной ленте.

Пример 2. Пусть задан последовательный набор данных с номером 5, содержащий две физические записи. Первая запись

содержит символы:

_15100__5010__8000_____50_____1

Вторая — символы:

_____1_____10_____50__1000_10000

В результате выполнения программы

```
      DIMENSION C(5),A(5),B(5)
    1 FORMAT ('1',3X,'УМНОЖЕНИЕ__ВЕКТОРОВ__
     *                  C=A*B'///'_C=')
    2 FORMAT (5F6.2)
    3 FORMAT ('+',T4,4(F5.2,','),F5.2';')
      READ (5,2)A,B
      WRITE (6,1)
      DO 7 I = 1,5
    7 C(I)=A(I)*B(I)
      WRITE (6,3)C
      RETURN
      END
```

на печать (в набор с номером 6) будут выданы строки символов, приведенные на рис. 8.2.

			У	М	Н	О	Ж	Е	Н	И	Е		В	Е	К	Т	О	Р	О	В			С	=	А	*	В			
С	=		1	.	5	1	,		5	.	0	1	,	4	0	.	0	0	,		5	.	0	0	,		1	.	0	0	;

Рис. 8. 2. Второй пример размещения данных на бумажной ленте.

Пример 3. Пусть задана программа:

```
    1 FORMAT ('1',15X,27('—')/16X,'I',12X,'I',12X,'I'/16X,
     *        'I',6X,'1X',5X,'I',4X,'F(X)',4X,'I'/16X,'I',12X,'I',
     *        12X,'I'/16X,27('—'))
    2 FORMAT (16X,'I_',F10.3'_I_',F10.3,'_I')
    4 FORMAT(16X,27('_'))
      WRITE(6,1)
      X=0.0
      DO 3 K = 1,5
      X=X+1.0
      Y=X**2+1.0
    3 WRITE(6,2)X,Y
```

```
      WRITE (3,4)
      RETURN
      END
```

В результате выполнения этой программы на печать (в набор с номером 6) будут выданы строки символов, приведенные на рис. 8.3.

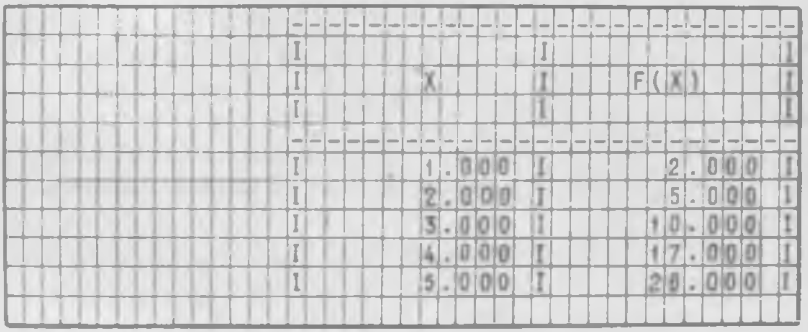


Рис. 8. 3. Третий пример размещения данных на бумажной ленте.

Вопросы и упражнения

1. Пусть задана программа:

```
      DIMENSION N(5),X(5)
    1 FORMAT('1',МАССИВ_ X'//('_ X',I1,'=',F5.2))
      DO 3 I=1,5
      X(I)=I+0.25
    3 N(I)=I
      WRITE(6,1)(N(I),X(I),I=1,5)
      STOP
      END
```

содержащая оператор вывода данных на бумажную ленту через АЦПУ (WRITE).

В каком виде данные будут представлены на бумажной ленте, если устройство печати работает в режиме с управляющими символами?

2. Пусть задана программа:

```
    1 FORMAT('1',25('—')/2H_ I,6X,'X',6X,'I',4X,'Y',4X,'I'/'
     _ ',25('—')/('_ I',1PE11.3,2X,'I',F7.1,2X,'I'))
    2 FORMAT ('_ ',25('—'))
      DIMENSION X(5),Y(5)
      B=0.001
```

```
      X(1)=0.5
      DO 5 I=1,5
      B=B*10.0
      Y(I)=B
    5 X(I)=(X(I)**2+0.005)*Y(I)
      WRITE (6,1)(X(I),Y(I),I=1,5)
      WRITE (6,2)
      RETURN
      END
```

содержащая операторы вывода данных на бумажную ленту через АЦПУ (операторы WRITE).

В каком виде данные будут представлены на бумажной ленте, если устройство печати работает в режиме с управляющими символами?

3. Пусть вещественные переменные S, S1, R, R1 имеют следующие значения:

$$S = 5170.4, \quad S1 = 2727.2, \quad R = -0.0000023, \quad R1 = 0.0000452$$

Написать фрагмент программы, обеспечивающий вывод на бумажную ленту через АЦПУ (через набор данных последовательного доступа со ссылочным номером 6) следующих данных:

а) последовательности символов ЗНАЧЕНИЯ ПЕРЕМЕННЫХ S, R, S1, R1, которые должны поместиться в первой строке страницы;

б) значений переменных S и R, которые должны поместиться в третьей строке страницы в формате G15.4;

в) значений переменных S1 и R1, которые должны поместиться в пятой строке страницы в формате G15.4.

Фрагмент программы составлять, исходя из предположения, что АЦПУ работает в режиме с управляющими символами.

4. Пусть значения вещественных переменных A, B, C, D нанесены на перфокарты в формате F7.3 таким образом, что каждое из них размещается на отдельной перфокарте, начиная с первой позиции. Составить программу, обеспечивающую ввод значений переменных с перфокарт (из набора данных последовательного доступа со ссылочным номером 5) и их вывод на бумажную ленту через АЦПУ (в набор данных последовательного доступа со ссылочным номером 6) в виде двух строк символов. Первая строка должна содержать символы $A = \alpha$; $B = \beta$; вторая строка — символы $C = \gamma$; $D = \varphi$; здесь через α, β, γ и φ обозначены значения переменных A, B, C и D соответственно, которые должны быть представлены в формате E12.5. Программу составлять, исходя из предположения, что АЦПУ работает в режиме с управляющими символами.

Глава 9

УПРАВЛЕНИЕ ВЫПОЛНЕНИЕМ ПРОГРАММ В ОПЕРАЦИОННОЙ СИСТЕМЕ ОС ЕС ЭВМ

9.1. Этапы подготовки фортран-программ к выполнению на ЭВМ

Программа, написанная на языке фортран, может выполняться на ЭВМ только лишь после ее перевода на язык этой ЭВМ. В операционной системе ОС ЕС ЭВМ такой перевод осуществляется в два этапа, к которым относятся *трансляция* и *редактирование*.

Этап трансляции выполняется на ЭВМ по программе, называемой *транслятором*. На этом этапе решаются основные вопросы, связанные с переводом фортран-программы на язык ЭВМ. В процессе трансляции каждый модуль фортран-программы переводится в так называемый *объектный модуль*.

Обычно исходные модули фортран-программы вводятся для обработки с перфокарт, а объектные модули помещаются на магнитные диски или магнитные ленты в наборы данных последовательного доступа.

Этап редактирования выполняется на ЭВМ по программе, называемой *редактором связей*. На этом этапе решаются две основные задачи: 1) объединение объектных модулей программы в единую программу, называемую *загрузочным модулем*; 2) включение в полученный загрузочный модуль ряда модулей из системной библиотеки фортрана для реализации математических функций (синуса, косинуса, логарифма и т. д.) и операторов ввода-вывода.

Исходные объектные модули берутся редактором связей из тех наборов, в которые их помещает транслятор, а загрузочные модули помещаются в личные или временные библиотеки программы, которые организуются на магнитных дисках.

Вопросы и упражнения

1. Назвать этапы перевода фортран-программ на язык ЭВМ.

2. Назвать основные задачи, которые решаются на этапе трансляции программ и на этапе их редактирования.

3. Как называются программные модули, полученные в результате работы транслятора и куда они обычно помещаются?

4. Как называются программные модули, полученные в результате работы редактора связей и куда они обычно помещаются?

9.2. Библиотеки программ

Операционная система ОС ЕС ЭВМ требует, чтобы все программы, подлежащие выполнению на ЭВМ, находились в библиотеке программ в виде загрузочных модулей.

Библиотеки программ размещаются в наборах данных с библиотечной организацией, которые могут создаваться только на устройствах прямого доступа (магнитных дисках).

Библиотечный набор состоит из двух частей: из *поля оглавления* и совокупности *полей данных*. Данные в библиотечном наборе размещаются группами, называемыми *разделами*. Каждому разделу присваивается имя (идентификатор, содержащий от 1 до 8 символов).

Оглавление и разделы можно рассматривать как наборы данных с последовательной организацией. Блоки оглавления имеют формат VB, максимальная длина блока составляет 256 байтов, а максимальная длина физической записи — 74 байта.

Каждая физическая запись оглавления предназначена для хранения справочной информации об одном разделе данных. В состав этой информации включаются такие сведения, как имя раздела, относительный адрес первой записи этого раздела внутри набора и т. д. Блоки всех разделов должны иметь одинаковые характеристики (формат, максимальную длину и т. д.), которые указываются в управляющих операторах задания на создание библиотечного набора.

Если библиотечный набор данных используется в качестве библиотеки программ (загрузочных модулей), то каждый загрузочный модуль занимает в нем отдельный раздел. При этом имя раздела считается именем загрузочного модуля. Поиск загрузочного модуля осуществляется операционной системой по имени библиотеки и по имени раздела, в которых содержится данный модуль.

В операционной системе ОС ЕС ЭВМ предусмотрено три типа библиотек загрузочных модулей: *общая* (*системная*), *личные* и *временные*.

Общая библиотека имеет имя SYS1.LINKLIB, в которой размещаются такие программы, как трансляторы с различных языков программирования (в том числе и с фортрана), редактор связей и др. обслуживающие программы. Эта библиотека поставляется вместе с операционной системой и всегда готова к использованию. Личные библиотеки создаются пользователями и предназначаются для хранения программ, разработанных пользователями. Временные библиотеки создаются и используются только в процессе выполнения одного задания (понятие задания будет описано в последующих разделах главы). Временные библиотеки, созданные в процессе выполнения задания, после его выполнения уничтожаются операционной системой без дополнительных указаний. Временные библиотеки обычно используются в процессе отладки программ.

Личные и временные библиотеки создаются редактором связей в процессе подготовки программ к выполнению на ЭВМ.

Вопросы и упражнения

1. Назвать виды библиотек загрузочных модулей. Для чего они предназначены?

2. В какой библиотеке хранятся загрузочные модули транслятора и редактора связей?

3. Как попадают программы в личные библиотеки загрузочных модулей?

4. На каких внешних запоминающих устройствах могут создаваться библиотеки загрузочных модулей?

9.3. Задания на выполнение программ

Выполнение програми на ЭВМ, оснащенных операционной системой ОС ЕС ЭВМ, осуществляется под управлением этой системы. Чтобы операционная система могла выполнить программу, ей необходимо сообщить ряд сведений, таких как имя программы, имя библиотеки, содержащей эту программу, шифры устройств (магнитных дисков, магнитных лент, устройств ввода данных с с перфокарт и т. д.), на которых размещаются используемые в программе наборы данных, характеристики наборов данных и т. д.

Совокупность сведений, необходимых операционной системе для выполнения одной программы или некоторой последовательности программ, называется *заданием*, а язык, на котором эти сведения записываются,— *языком управления заданиями*.

Одно задание может содержать указания на выполнение одной или нескольких программ (загрузочных модулей). Совокупность

сведений, необходимых для выполнения одной программы, называется *шагом задания*.

Обычно шаги задания выполняются в той последовательности, в которой они располагаются в задании. Однако в языке управления заданиями имеются средства, позволяющие описать условия пропуска того или иного шага задания.

Операционная система вводит задания для обработки порциями, которые принято называть *пакетами заданий*. Задания внутри одного пакета считаются независимыми единицами работы и могут обрабатываться в произвольном порядке. Порядок их обработки зависит от варианта операционной системы (PCP, MFT или MVT).

Операционная система, работающая в режиме PCP (однопрограммном режиме), ведет обработку заданий в порядке их следования в пакете. В операционной системе, работающей в режиме MFT (мультипрограммном режиме с фиксированным количеством задач) и MVT (мультипрограммном режиме с переменным числом задач), порядок обработки заданий зависит от их класса и приоритета.

Задания вводятся в ЭВМ через устройство, которое в операционной системе ОС ЕС ЭВМ имеет статус *системного устройства ввода*.

В качестве такого устройства обычно используется устройство ввода с перфокарт. Пакеты заданий, вводимые в ЭВМ через системное устройство ввода, часто называют *входным потоком*. Это понятие используется в описании языка управления заданиями.

Ниже приводится описание упрощенного варианта языка управления заданиями, позволяющего составлять задания на выполнение большинства действий, связанных с подготовкой фортран-программ к выполнению на ЭВМ, а также с их выполнением.

Вопросы и упражнения

1. Что представляет собой задание на выполнение программ?

2. На каком языке составляются задания на выполнение программ?

3. Где хранятся программы, вызываемые операционной системой на исполнение?

4. Что представляет собой пакет заданий?

5. Описать порядок выполнения задач, содержащихся в задании.

6. Описать порядок выполнения заданий, содержащихся в пакете.

9.3.1. Структура задания. Задание составляется из *управляющих операторов* (предложений).

Для ввода в ЭВМ управляющие операторы, составляющие задание, наносятся на перфокарты. Перфокарты с нанесенными на них управляющими операторами принято называть *управляющими картами*.

Существуют пять типов управляющих операторов, основными из которых являются *оператор начала задания* (JOB), *оператор начала шага* (EXEC), *оператор описания набора данных* (DD) и *оператор конца задания*. Остальные управляющие операторы служат для описания процедур и используются редко.

Задание должно начинаться с оператора JOB, с помощью которого операционной системе сообщаются сведения, относящиеся ко всему заданию. За этим оператором может следовать оператор DD, который описывает личную библиотеку, называемую *библиотекой задания*. Далее должны следовать шаги задания. Каждый шаг задания должен начинаться с оператора EXEC, с помощью которого операционной системе сообщаются сведения о программе, выполняемой на данном шаге. За оператором EXEC может следовать оператор DD, описывающий личную библиотеку, называемую библиотекой шага задания. Далее должны следовать операторы DD, описывающие наборы данных, используемые в программе данного шага.

Опишем порядок поиска программы, указанной в операторе EXEC, в библиотеках загрузочных модулей.

Случай 1. В задании не описаны ни библиотека задания, ни библиотека шага задания. Тогда поиск программы будет осуществляться только в общей библиотеке с именем SYS1.LINKLIB.

Случай 2. В задании описана только библиотека задания. Тогда поиск программы будет осуществляться сначала в библиотеке задания, а затем в общей библиотеке системы с именем SYS1.LINKLIB.

Случай 3. В рассматриваемом шаге задания описана библиотека шага. Тогда поиск программы будет осуществляться сначала в библиотеке этого шага задания, а затем в общей библиотеке системы с именем SYS1.LINKLIB. В этом случае библиотека задания игнорируется.

Случай 4. В задании описана библиотека задания, в некоторых шагах задания описаны библиотеки шагов задания, однако в рассматриваемом шаге библиотека шага не описана. Тогда поиск программы будет осуществляться сначала в библиотеке задания, а затем в общей библиотеке системы с именем SYS1.LINKLIB.

Задание на выполнение программ пишется на специальных бланках. Каждая строка бланка содержит такое же количество

колонок, что и перфокарта (80 колонок). Строка начинается с идентифицирующих символов // (две косые черточки), которые пишутся в первой и второй колонках. В последующих колонках размещаются поля управляющих операторов.

Каждый управляющий оператор может содержать до четырех полей, которые имеют следующие названия: *поле имени*, *поле операции*, *поле параметров*, *поле комментария*.

Поле имени — это идентификатор, содержащий от одного до восьми символов. Оно размещается в строке, начиная с третьей колонки. В некоторых операторах поле имени может отсутствовать.

В поле операции указывается код операции (тип) управляющего оператора, который определяет его назначение. Поле операции пишется на бланке следом за полем имени (а при его отсутствии следом за полем идентифицирующих символов), отделяясь от него одним или несколькими пробелами. Это поле является обязательным для всех управляющих операторов, кроме оператора конца задания, который содержит только идентифицирующие символы //. В поле параметров указывается список параметров, посредством которых операционной системе передаются различные сведения. Поле параметров размещается следом за полем операции и отделяется от него одним или несколькими пробелами. Параметры в списке отделяются друг от друга запятыми.

Управляющие операторы могут содержать параметры двух типов: *позиционные* и *ключевые*.

Позиционные параметры размещаются в поле параметров первыми и в строго определенном порядке. Эти параметры имен не имеют, и в поле параметров помещаются только их значения. При написании управляющих операторов некоторые из позиционных параметров разрешается опускать. В этом случае запятые, отделяющие эти параметры от остальных, нужно сохранять. Так, например, управляющий оператор может иметь вид:

$//i\ p\ a_1,a_2,,,a_5$

где i — имя оператора; p — код операции; a_1 — значение первого позиционного параметра; a_2 — значение второго позиционного параметра; a_5 — значение пятого позиционного параметра. Третий и четвертый параметры опущены.

Если опускаются один или несколько последних (по порядку их следования) позиционных параметров, то разделяющие их запятые не указываются. Так, например, предыдущий управляющий оператор может иметь вид:

$//\ i\ p\ a_1,,a_2$

(опущены второй, четвертый и пятый параметры).

Позиционный параметр может представлять собой список позиционных и ключевых подпараметров. В этом случае список подпараметров заключается в круглые скобки, а подпараметры отделяются друг от друга запятыми. Правила размещения подпараметров внутри списка такие же, как и правила размещения параметров в поле параметров. Так, например, позиционный параметр может иметь вид:

$(b_1,,b_3,b_4)$

где b_1 — значение первого подпараметра; b_3 — значение третьего подпараметра; b_4 — значение четвертого подпараметра. Второй подпараметр опущен.

Ключевые параметры размещаются в поле параметров следом за позиционными параметрами. Относительно друг друга эти параметры могут следовать в произвольном порядке.

Ключевой параметр имеет вид:

$a = b$

где a — имя ключевого параметра, b — либо значение ключевого параметра, либо список подпараметров, заключенный в круглые скобки. Так, например, ключевой параметр может иметь вид:

$x = (x_1, x_2, y = y_1)$

где x — имя параметра, x_1 — значение первого (позиционного) подпараметра, x_2 — значение второго (позиционного) подпараметра, y — имя ключевого подпараметра, y_1 — значение ключевого подпараметра. Заметим, что подпараметр в свою очередь может иметь несколько подпараметров и т. д.

Поле комментария — это последовательность произвольных символов. Оно следует за полем параметров, отделяясь от него одним или несколькими пробелами. Комментарий операционной системой не используется.

Управляющий оператор может размещаться на бланке, начиная с первой колонки и кончая 71-й колонкой. Если он не умещается в одной строке, то может быть перенесен в последующие строки. При этом должны соблюдаться следующие *правила переноса*:

1) Поле имени и поле кода операции на следующую строку не переносятся.

2) Поле операторов можно прерывать для переноса на другую строку только после запятой, отделяющей параметры или подпараметры друг от друга. При этом в 72-й колонке бланка

может записываться знак переноса (любой символ, отличный от пробела).

3) Строка продолжения должна начинаться с идентифицирующих символов //.

Начало перенесенной части должно отделяться от поля идентифицирующих символов одним или несколькими пробелами, причем пробелов должно быть не более тринадцати.

4) Поле комментария может прерываться для переноса на другую строку в любом месте. В этом случае знак переноса указывается обязательно. Начало перенесенной части от поля идентифицирующих символов может ничем не отделяться.

Комментарий может быть оформлен и в виде самостоятельного оператора, который может размещаться за любым управляющим оператором. В качестве имени этого оператора используется символ * (звездочка), который помещается следом за символами //. Текст может размещаться в колонках с 4-й по 80-ю, в нем могут использоваться любые символы. Оператор комментария может размещаться только в одной строке.

Ввод пакетов заданий в ЭВМ осуществляется через *входное системное устройство*. Выдача сообщений о ходе выполнения заданий осуществляется на одно из *выходных системных устройств* (обычно на АЦПУ).

Каждому из выходных системных устройств ставится в соответствие определенный *выходной класс*. Выходные классы обозначаются латинскими заглавными буквами (от A до Z) и десятичными цифрами (от 0 до 9).

В ЭВМ, имеющих небольшое количество внешних устройств, обычно предусматриваются два выходных системных устройства — это АЦПУ и устройство вывода на перфокарты. Первому из них приписывается класс A, а второму — класс B.

Данные, выдаваемые на системные выходные устройства, сначала поступают на магнитные ленты или на магнитные диски, где они образуют так называемые *выходные потоки* (*выходные очереди*), которые после выполнения задания разгружаются (выдаются на соответствующие выходные устройства) специальной *программой разгрузки*. Запуск программы разгрузки осуществляется операторами с пульта управления ЭВМ.

Работа АЦПУ и устройства выдачи на перфокарты в режиме с выходными очередями обеспечивает эффективное использование этих устройств.

Этот режим находит широкое применение не только в системных программах, но и в программах пользователей.

Вопросы и упражнения

1. Назвать основные типы операторов языка управления заданиями.

2. Описать структуру управляющего оператора.

3. Какие типы параметров используются в управляющих операторах задания?

4. Описать правило переноса управляющих операторов со строки на строку.

9.3.2. Оператор начала задания (JOB). Оператор JOB в общем случае имеет вид:

//*i* JOB a_1,a_2,MSGLEVEL=b_1,COND=b_2,

// TIME=b_3,CLASS=b_4,PRTY=b_5,

// MSGCLASS=b_6,REGION=b_7

где *i* — имя управляющего оператора; a_1, a_2 — значения позиционных параметров; b_1, b_2, . . ., b_7 — значения ключевых параметров; MSGLEVEL, COND и т. д.— имена ключевых параметров.

Имя оператора JOB выбирается разработчиком программы и должно присутствовать обязательно. Оно воспринимается операционной системой как имя всего задания. Для операционных систем, работающих в режимах MFT и MVT, имена заданий, входящих в один пакет, должны быть различными. Поле параметров может отсутствовать. Ниже приводятся описания параметров.

1) Позиционные параметры.

Параметр a_1 содержит учетный номер задания, а параметр a_2 — сведения о разработчике. Вид этих параметров зависит от программ учета заданий. Эти программы разрабатываются и включаются в операционную систему вычислительными центрами. Если учетная информация содержит внутри себя запятые, то она должна заключаться либо в круглые скобки, либо в апострофы. Учетная информация может содержать не более 142 символов. Сведения о разработчике могут содержать не более 20 символов. Если в операционной системе программы учета нет, то эти параметры опускаются. В последующих примерах параметры a_1 и a_2 указываться не будут.

2) Параметр MSGLEVEL.

При выполнении задания на печать могут выдаваться следующие сведения: операторы задания и сообщения об имеющихся в них синтаксических ошибках; сведения о размещении описанных в задании наборов данных. Управление выдачей сведений осуществляется с помощью ключевого параметра, который в общем случае имеет

вид:

MSGLEVEL=(n_1,n_2)

где подпараметр n_1 служит для управления выдачей сведений, относящихся к операторам задания, а подпараметр n_2 — для управления выдачей сведений, относящихся к наборам данных. Подпараметр n_1 может принимать значения 0, 1 и 2, а подпараметр n_2 — значения 0 и 1.

Рассмотрим эти случаи.

а) Значение n_1 равно нулю. На печать выдаются первый управляющий оператор задания (оператор JOB) и сведения об ошибках в управляющих операторах.

б) Значение n_1 равно единице. На печать выдаются все управляющие операторы, включая и те, которые содержатся в указанных в задании каталогизированных процедурах. Сообщения об ошибках печатаются следом за теми операторами, в которых обнаружены эти ошибки.

в) Значение n_1 равно двум. На печать выдаются все управляющие операторы, однако содержимое каталогизированных процедур не печатается. Сообщения об ошибках печатаются следом за теми управляющими операторами, которые содержат эти ошибки.

г) Значение n_2 равно нулю. Сведения о размещении наборов данных выдаются только при аварийном завершении задания.

д) Значение n_2 равно единице. Сведения о размещении наборов данных выдаются всегда.

Приведем несколько примеров параметра MSGLEVEL:

а) MSGLEVEL=(2,0)

б) MSGLEVEL=(0,0)

в) MSGLEVEL=(1,1)

Стандартными значениями подпараметров являются 0 (для n_1) и 1 (для n_2). Если подпараметр имеет стандартное значение, то его можно опустить. Если оба подпараметра имеют стандартные значения, то можно опустить весь параметр.

Приведем несколько примеров сокращенной записи этого параметра:

а) MSGLEVEL=(,0)

б) MSGLEVEL=(1)

В первом из этих примеров считается опущенным первый подпараметр, а во втором — второй подпараметр.

Если в списке подпараметров остается только первый подпараметр, то скобки также можно опустить. Так, например, рассматриваемый нами параметр может иметь вид:

MSGLEVEL=1

В этом примере опущенным считается второй подпараметр.

3) Параметр COND.

Если в задании предусмотрено несколько шагов и выполнение последующих шагов зависит от результатов выполнения предыдущих, то эту зависимость можно указать в ключевом параметре, который имеет вид:

$$\text{COND}=((y_1),(y_2),\ldots,(y_n))$$

где y_i ($i = 1, 2, \ldots, n$) — условия окончания выполнения задания. Количество условий (n) может колебаться от 1 до 8. Если параметр COND содержит только одно условие, то он представляется в следующем виде:

$$\text{COND}=(y_1)$$

Условие имеет вид: α, σ, где α — целое число без знака (включая нуль); σ — один из знаков отношения: GT (больше), GE (больше или равно), LT (меньше), LE (меньше или равно), NE (не равно), EQ (равно).

Примеры записей параметра COND:

а) COND=(4,GT)

б) COND=((1,EQ),(3,LT),(5,GT))

Проверка условий осуществляется после выполнения каждого шага задания. Код завершения β, выработанный программой шага задания, подставляется во все условия таким образом, чтобы образовывались отношения вида: $\alpha_1\sigma_1\beta$, $\alpha_2\sigma_2\beta,\ldots,\alpha_n\sigma_n\beta$, где α_i и σ_i — элементы условий, входящих в состав параметра COND. Если хотя бы одно из образованных отношений удовлетворяется, то выполнение задания прекращается.

Приведем пример.

Пусть параметр COND имеет вид:

COND=((5,LT),(2,EQ))

и при выполнении программы первого шага задания выработался код завершения, равный двум. Из условий, входящих в параметр

COND, образуем отношения: 5LT2 и 2EQ2. Первое из этих отношений не удовлетворяется, а второе — удовлетворяется. Следовательно, выполнение задания будет прекращено после выполнения первого шага.

В программах, написанных на языке фортран код завершения β можно передать операционной системе через оператор останова, имеющий вид STOP β, где β — целое число без знака, которое передается операционной системе в качестве кода завершения.

Заметим, что операционной системе можно передавать коды завершения только из основной программы. Коды завершения из подпрограмм операционной системе не передаются. Если основная программа заканчивает свое выполнение оператором RETURN, то в качестве кода завершения передается число нуль.

4) Параметр TIME.

Этот параметр позволяет задавать отрезок времени работы центрального процессора, по истечении которого выполнение задания прекращается. Это делается главным образом с целью избежания зацикливания, вызываемого ошибками в программе или сбоями в оборудовании ЭВМ.

Параметр TIME имеет вид:

$$\text{TIME}=(n_1,n_2)$$

где n_1 означает минуты, а n_2 — секунды. Если $n_2 = 0$, то параметр TIME можно представлять в следующем виде:

$$\text{TIME}=n_1$$

Если параметр TIME отсутствует, то время работы центрального процессора будет определяться стандартным значением, которое назначается при генерации операционной системы.

Параметром TIME можно пользоваться только в операционных системах, работающих в режимах MFT и MVT.

5) Параметр CLASS.

Этот параметр имеет вид:

$$\text{CLASS}=a$$

где a — одна из 15-ти букв латинского алфавита A,B,...,O, означающая класс задания. Класс задания используется в системах, работающих в режимах MFT и MVT, для определения очередности выполнения заданий, содержащихся в пакете. Какие классы можно использовать и как они влияют на очередность выполнения заданий зависит от конкретной конфигурации операционной системы, используемой в вычислительном центре.

Если параметр CLASS не указан, то заданию назначается класс A.

6) Параметр PRTY. Этот параметр имеет вид:

PRTY=*n*

где *n* — целое число без знака, лежащее в пределах от 0 до 13 и означающее приоритет задания внутри своего класса. Чем выше приоритет (чем больше число *n*), тем раньше задание поступает на выполнение.

Приоритет 13 используется системными программами, и его назначать программам пользователей не рекомендуется. Стандартное значение приоритета назначается при генерации системы.

Приоритет используется только в операционных системах, работающих в режимах MFT и MVT.

7) Параметр MSGCLASS.

Этот параметр имеет вид:

MSGCLASS = *b*

где *b* — класс выходного системного устройства, на которое будет выдаваться информация о ходе выполнения задания.

Классы выходным системным устройствам назначаются в процессе генерации операционной системы. Стандартным значением этого параметра является класс A, которому обычно соответствует устройство построчной печати (АЦПУ).

Параметр MSGCLASS можно использовать только в операционных системах, работающих в режимах MFT и MVT.

8) Параметр REGION.

В операционной системе, работающей в режиме PCP, под задание выделяется вся оперативная память, за исключением той ее части, которая занимается самой операционной системой. В операционной системе, работающей в режиме MFT, объем выделяемой заданию памяти, определяется объемом того раздела памяти, в который это задание попадет для выполнения. Это зависит от класса задания.

Если система работает в режиме MVT, то запрос на выделение заданию оперативной памяти осуществляется с помощью параметра REGION, который имеет вид:

REGION=*n*K

где *n* — целое число без знака, не равное нулю. Заданию будет выделено *n*·1024 байтов памяти (1K=1024 байтам).

Если параметр REGION опущен, то заданию будет выделен стандартный объем памяти, который устанавливается при генерации системы.

В заключение данного раздела рассмотрим пример оператора JOB:

```
//FOR1 JOB MSGLEVEL=2,CLASS=F,
//  COND=(4,LT),TIME=15
```

В этом примере заданию присвоено имя FOR1. Поле операторов содержит 4 параметра. В процессе выполнения задания на печать будут выданы управляющие операторы без содержимого каталогизированных процедур, а также сведения о размещении наборов данных на внешних запоминающих устройствах. Заданию присвоен класс F. Если при выполнении какого-либо шага задания программа выдает код завершения, превышающий число 4, то выполнение задания будет прекращено. Выполнение задания прекращается и в том случае, если процессор затратит на выполнение задания больше 15-ти минут времени. Управляющие операторы и сведения о размещении наборов данных будут выдаваться на системное выходное устройство, соответствующее классу A.

Вопросы и упражнения

1. Перечислить обязательные элементы управляющего оператора JOB (элементы, которые должны присутствовать всегда).

2. Перечислить сведения, которые будут выдаваться операционной системой при выполнении некоторого задания, если оператор начала задания имеет вид:

а) //SIS1 JOB MSGLEVEL=(0,0)
б) //SIS2 JOB MSGLEVEL=(1,0)
в) //SIS3 JOB MSGLEVEL=(2,0)
г) //SIS4 JOB MSGLEVEL=(0,1)
д) //SIS5 JOB MSGLEVEL=(2,1)
е) //SIS6 JOB MSGLEVEL=(,1)
ж) //SIS7 JOB MSGLEVEL=2
з) //SIS8 JOB

3. Пусть задание содержит несколько шагов. Шаги второй, третий и т. д. должны исполняться только тогда, когда ни на одном из предыдущих шагов не был выработан код завершения, превышающий число 8. Написать оператор начала задания, обеспечивающий указанный режим выполнения задания.

4. Пусть имеется задание, состоящее из нескольких шагов. Написать оператор начала задания, обеспечивающий следующий режим его выполнения:

а) в процессе выполнения задания на печатающее устройство должны выдаваться все управляющие операторы, а также сведения о размещении наборов данных на внешних запоминающих устройствах; устройство печати имеет класс A;

б) первый шаг задания должен выполняться всегда;

в) второй и последующие шаги задания должны выполняться только в том случае, если ни на одном из предыдущих шагов не был выработан код завершения K=4;

г) выполнение задания должно быть прекращено, если на него будет затрачено более 25 минут времени работы центрального процессора;

д) задание должно иметь класс B;

е) заданию должен выделяться стандартный объем оперативной памяти.

9.3.3. Оператор начала шага задания (EXEC). Существует две формы представления оператора EXEC. Первая из них используется для непосредственного вызова программ на исполнение, а вторая — для вызова программ на исполнение с помощью каталогизированных процедур. В данном разделе мы познакомимся с первой из этих форм.

Оператор EXEC имеет следующий вид:

```
//i EXEC PGM=b1,PARM=b2,COND=b3,
//  TIME=b4,REGION=b5
```

где i — имя оператора; $b_1, b_2, \ldots, b_5$ — значения ключевых параметров; PGM, PARM и т. д.— имена ключевых параметров.

Имя оператора должно присутствовать только в том случае, если в последующих шагах задания имеются ссылки на этот оператор. Параметр PGM является обязательным, и в поле параметров он должен размещаться на первом месте. Ниже приводятся описания параметров оператора EXEC.

1) Параметр PGM.

Этот параметр имеет вид

```
PGM = u
```

где u — либо имя программы, либо ссылка на описание временной библиотеки, в которую помещается программа редактором связи в одном из предыдущих шагов задания.

Имя программы указывается в том случае, если программа, подлежащая исполнению, переведена на машинный язык заранее и содержится в библиотеке программ (загрузочных модулей).

Так, например, чтобы оттранслировать программу, написанную на языке фортран, нужно из системной библиотеки загрузочных модулей SYS1.LINKLIB вызвать транслятор, который имеет имя IEYFORT. В этом случае параметр PGM будет выглядеть так:

PGM=IEYFORT

Если же программа, которую необходимо выполнить, готовится для выполнения (транслируется и редактируется) в этом же задании (в предыдущих шагах) и редактор связей помещает ее во временную библотеку, то параметр PGM должен иметь вид

PGM=*.u_1.u_2

где u_1 — имя того оператора EXEC, который используется для вызова редактора связей, u_2 — имя оператора DD, который описывает временную библиотеку как выходной набор данных для редактора связей.

Рассмотрим следующий пример: пусть требуется выполнить программу, которую редактор связей в предыдущем шаге задания записал во временную библиотеку. Оператор EXEC, вызывающий редактор связей, имеет имя LKED, а оператор DD, описывающий временную библиотеку, имеет имя SYSLMOD. Тогда вызов этой программы можно осуществить оператором EXEC, имеющим вид:

// EXEC PGM=*.LKED.SYSLMOD

2) Параметр PARM.

Этот параметр имеет вид:

PARM = n

где n — последовательность символов, которая передается вызванной из библиотеки программе. Количество передаваемых символов, их структура и назначение определяются той программой, которая их использует.

Это средство широко используется для управления работой таких системных программ, как трансляторы, редактор связей и др.

Программы, составляемые на языке фортран, не могут использовать данные, передаваемые им через параметр PARM.

3) Параметр COND.

Если в задании предусмотрено несколько шагов и выполнение некоторого шага зависит от результатов предыдущих шагов, то эту

зависимость можно указать в ключевом параметре COND, входящем в состав оператора EXEC.

Параметр COND имеет вид

$$\text{COND}=((y_1),(y_2),\ldots,(y_n))$$

где y_i $(i = 1, 2,\ldots,n)$ — условие пропуска данного шага задания. Количество условий не должно превосходить восьми. Если требуется указать только одно условие, то этот параметр представляется без внешних скобок: COND=(*y*).

Условия, указанные в параметре COND, проверяются перед выполнением шага задания. Если хотя бы одно условие выполняется, то текущий шаг задания не выполняется (осуществляется переход к следующему шагу задания).

Основная форма записи условия такова: α, σ, *j*, где α — целое число без знака (включая нуль), σ — один из знаков отношения (GT, GE, LT, LE, EQ, NE), *j* — имя оператора EXEC, относящегося к одному из предыдущих шагов задания (имя шага задания).

Условие считается выполненным, если удовлетворяется отношение $\alpha\sigma\beta$, где β — код завершения, выработанный на *j*-ом шаге задания.

Подпараметр *j* может отсутствовать. В этом случае условие считается выполненным, если удовлетворяется хотя бы одно из отношений $\alpha\sigma\beta_1, \alpha\sigma\beta_2, \ldots, \alpha\sigma\beta_k$. Здесь β_1 — код завершения, выработанный на первом шаге задания, β_2 — код завершения, выработанный на втором шаге задания, β_k — код завершения, выработанный на *k*-ом (т. е. на предыдущем) шаге задания.

Если в параметре COND условия представлены в рассмотренной выше форме, то в случае аварийного завершения хотя бы одного из предыдущих шагов задания, данный шаг будет пропущен (независимо от выполнения содержащихся в параметре COND условий).

Чтобы этого избежать, достаточно первое (или единственное) условие параметра COND представить в виде слова EVEN. Так, например, параметр COND может иметь вид

COND=(EVEN,(8,EQ)).

В этом случае аварийное завершение одного или нескольких предыдущих шагов не будет влиять на выполнение текущего шага.

Если первое (или единственное) условие параметра COND представить в виде слова ONLY, то шаг задания, содержащий данный параметр, будет выполняться только в том случае, если произойдет аварийное завершение хотя бы одного из предыдущих

шагов задания. Остальные условия влияют на выполнение шага обычным образом.

4) Параметр TIME.

Этот параметр можно использовать только при работе операционной системы в режимах MFT и MVT. Он кодируется таким же образом, что и параметр TIME в операторе JOB. В этом параметре указывается суммарное время, которое центральный процессор может затратить на выполнение данного шага задания. Если время процессора превысит указанный интервал времени, то шаг задания будет завершен аварийно.

Если параметр TIME указан и в операторе JOB, и в операторе EXEC, то значение параметра TIME в операторе EXEC не должно превышать значения параметра TIME в операторе JOB.

5) Параметр REGION.

Этот параметр кодируется таким же образом, что и параметр REGION в операторе JOB. С помощью параметра REGION осуществляется запрос на выделение данному шагу задания необходимой порции оперативной памяти. Если этот параметр указан и в операторе JOB, и в операторе EXEC, то оперативная память выделяется на все задание (параметр REGION в операторе EXEC игнорируется).

Примеры операторов EXEC:

```
//ST1 EXEC PGM=TASK
//ST2 EXEC PGM=POG,COND=EVEN,TIME=10
//     EXEC PGM=IEYFORT,PARM=(LIST,MAP)
```

Вопросы и упражнения

1. Перечислить обязательные элементы управляющего оператора EXEC.

2. Пусть в некоторой библиотеке содержится программа с именем PROG1. Написать оператор начала шага задания, обеспечивающий вызов этой программы на исполнение.

3. Пусть некоторое задание содержит управляющий оператор

```
//  EXEC PGM=FORT1,COND=(4,EQ),TIME=15,
//           REGION=7K
```

Ответить на следующие вопросы:

а) Какое имя имеет программа, вызываемая на исполнение управляющим оператором EXEC?

б) Как зависит исполнение этой программы от программ, исполняемых в других (предыдущих) шагах задания?

в) Какой объем оперативной памяти будет выделен этой программе, если операционная система будет работать в режиме MVT?

9.3.4. Оператор описания наборов данных последовательного и прямого доступов (DD). 1) *Общий вид оператора описания наборов данных.*

Оператор описания набора данных имеет вид:

$$//i \text{ DD } p$$

где i — имя оператора; p — совокупность параметров.

Имя оператора должно присутствовать обязательно. Для наборов данных, используемых в фортран-программах, оно должно иметь вид:

$$\text{FT}c_1c_2\text{F001}$$

где c_1c_2 — ссылочный номер набора данных, состоящий из двух десятичных цифр.

Выбор устройства для размещения того или иного набора данных, вообще говоря, не зависит от ссылочного номера, присвоенного этому набору. Однако, описание управляющих операторов упрощается, если использовать принятые в трансляторе стандартные назначения устройств и стандартные характеристики наборов данных.

Так, например, в трансляторе уровня G приняты следующие стандартные назначения устройств: наборам данных со ссылочными номерами с 1 по 4 и с 8 по 99 назначаются устройства типа магнитных лент и пакетов магнитных дисков; набору данных с номером 5 — устройство ввода с перфокарт; набору данных с номером 6 — устройство выдачи на построчную печать (АЦПУ), набору данных с номером 7 — устройство выдачи на перфокарты.

Если в управляющих операторах DD, описывающих наборы данных последовательного доступа, не указываются значения таких характеристик, как форматы и размеры блоков, то им присваиваются стандартные значения. Стандартные характеристики блоков для наборов данных последовательного доступа применительно к транслятору уровня G приведены в табл. 9.1.

Буква A в формате блока указывает на использование режима работы с управляющими символами.

Необходимо заметить, что набор данных со ссылочным номером 6 используется модулем ввода-вывода, включаемым редактором связи во все фортран-программы, для выдачи сообщений об ошибках, выявляемых в процессе выполнения программы. Поэтому шаги задания, предназначенные для вызова программ, составленных на

Таблица 9.1. Стандартные характеристики наборов данных последовательного доступа для транслятора уровня G

Ссылочный номер набора	Имя оператора	Размер блока в байтах	Формат блока	
			Форматный ввод-вывод	Бесформатный ввод-вывод
1	FT01F001	800	U	VS
2	FT02F001	800	U	VS
3	FT03F001	800	U	VS
4	FT04F001	800	U	VS
5	FT05F001	80	F	VS
6	FT06F001	129	UA	VSA
7	FT07F001	80	F	VS
8	FT08F001	800	U	VS
.	.	.	.	.
.	.	.	.	.
.	.	.	.	.
99	FT99F001	800	U	VS

языке фортран, обязательно должны иметь описания наборов со ссылочным номером 6.

Вид параметров оператора DD в сильной степени зависит от типа запоминающего устройства, на котором размещается описываемый набор данных. Поэтому дальнейшее описание элементов управляющего оператора DD будет вестись применительно к конкретным устройствам.

В процессе описания управляющих операторов DD используется такое понятие, как код устройства. В качестве кода устройства могут указываться *физический адрес устройства*, *типовое имя устройства* или *групповое имя устройства*.

Адрес устройства однозначно определяет конкретное устройство, на котором размещается набор данных. Он представляется в виде трехзначного целого числа без знака в шестнадцатеричной системе счисления. Типовое имя определяет собой группу однотипных устройств и представляется в виде четырехзначного числа без знака. Групповое имя определяет группу устройств, в которую могут входить разнотипные устройства. Типовые и групповые имена обычно используются тогда, когда системе предоставляется право выбора конкретного устройства для размещения на нем набора данных. Выбор осуществляется среди устройств, объединенных указанным типовым или групповым именем.

Коды устройств (физические адреса, типовые и групповые имена) устанавливаются в процессе генерации системы.

Ниже приводятся примеры групповых и типовых имен, используемых в операционной системе ОС ЕС ЭВМ:

SYSSQ — групповое имя магнитофонов и дисководов всех типов;

SYSDA — групповое имя дисководов всех типов;

5050 — типовое имя (номер) дисководов для пакетов на 7,25 мегабайт;

5061 — типовое имя (номер) дисководов для пакетов на 29 мегабайт;

5010 — типовое имя (номер) для магнитофонов.

2) Ссылка на полное описание набора данных путем использования параметра DDNAME.

Если в шаг задания включен оператор DD, содержащий полное описание некоторого набора данных, и по каким-то причинам в него нужно включить второй оператор DD, содержащий точно такое же описание набора данных, то второй оператор можно представить в сокращенном виде:

```
//j DD DDNAME = i
```

где j — имя оператора DD; i — имя первого оператора DD, содержащего полное описание набора данных.

3) Игнорирование операций ввода-вывода путем использования параметра DUMMY.

Если оператор DD, описывающий набор данных последовательного доступа, представить в следующем виде:

```
//j DD DUMMY
```

то операции ввода-вывода, относящиеся к этому набору, выполняться не будут.

Вопросы и упражнения

1. Каким образом ссылочный номер набора данных, используемого фортран-программой, связан с именем оператора DD, содержащего описание этого набора?

2. Назвать стандартные характеристики наборов данных последовательного доступа для транслятора уровня G.

3. Написать оператор DD, который подавляет действия операторов ввода-вывода в процессе их обращения к набору данных со ссылочным номером 6.

9.3.5. Оператор DD для входного перфокарточного набора данных. Входной перфокарточный набор данных представляет собой колоду перфокарт с нанесенными на них данными. Он относится к наборам последовательного доступа, блоки которого могут иметь форматы F или FB. Блоки таких наборов могут содержать только по одной физической записи. Обращение к перфокарточному набору может осуществляться только через операторы форматного ввода-вывода. В этом наборе под блок (физическую запись) отводится либо целая перфокарта, либо ее часть. Признак конца набора данных (символы /*) помещается в первой и второй колонках последней перфокарты.

Устройство ввода данных с перфокарт не обеспечивает возврата набора данных на предыдущую запись или на начало набора, поэтому при работе с таким устройством операторы BACKSPACE и REWIND не вызывают никаких действий.

Имя набора на перфокарты не наносится. Установка набора на первый блок осуществляется человеком, обслуживающим ЭВМ. Управляющий оператор DD для входного перфокарточного набора данных может представляться в одной из следующих форм:

1) //*i* DD UNIT=a_1,DCB=(RECFM=a_2,
   // BLKSIZE=a_3, LRECL=a_4)

2) //*i* DD DATA

Здесь приняты следующие обозначения: *i* — имя оператора DD; a_1 — код устройства ввода данных с перфокарт; a_2 — формат блоков набора данных; a_3 — длина блоков набора данных; a_4 — длина физических записей.

Форма 1 используется в том случае, если ввод данных осуществляется через устройство, которое не определено как системное входное устройство (т. е. не предназначено для ввода заданий). В этом случае в качестве кода устройства ввода с перфокарт обычно используется его физический адрес. Форму 1 можно использовать для описания набора данных с любым ссылочным номером.

При использовании формата F длину физических записей можно не указывать. Если входной перфокарточный набор данных имеет стандартные характеристики (ссылочный номер 5, формат блоков F, длину блоков 80 байтов), то параметр DCB можно опустить.

Форма 2 используется при вводе перфокарт через системное устройство ввода. В этом случае перфокарты с нанесенными на них данными должны включаться в задание следом за оператором DD.

Считается, что такой набор имеет формат F и длину блоков 80 байтов.

Если ни одна из перфокарт набора данных в первой и второй колонках не содержит косых черточек (символов //), то вместо параметра DATA можно указать символ * (звездочка), что сокращает запись этого параметра.

Форму 2 и ее сокращенный вариант можно использовать для описания входного набора данных с любым ссылочным номером.

Заметим, что форма 1 используется редко. Это вызвано двумя обстоятельствами. Во-первых, вычислительные установки часто снабжаются только одним устройством ввода с перфокарт, которое используется как системное входное устройство (для ввода заданий). Во-вторых, при работе операционной системы в режимах MFT и MVT очередность выполнения заданий, входящих в пакет, зависит от многих обстоятельств и предугадать ее практически невозможно. Это создает определенные трудности при определении очередности установки комплектов перфокарт на устройства ввода.

При использовании формы 2 следить за очередностью установки комплектов перфокарт не требуется.

Вопросы и упражнения

1. Через какое устройство осуществляется ввод данных с перфокарт?

2. Назвать стандартные характеристики входного перфокарточного набора данных.

3. Написать оператор DD для входного перфокарточного набора данных, имеющего ссылочный номер 5, формат F и длину блока, равную 80 байтам. Ввод данных должен осуществляться:

а) через системное устройство ввода с перфокарт;

б) через устройство ввода с перфокарт, которое не определено как системное и имеет физический адрес 183.

4. Выполнить упражнение 3, предварительно заменив в нем ссылочный номер 5 на ссылочный номер 8.

5. Выполнить упражнение 3, предварительно заменив в нем ссылочный номер 5 на ссылочный номер 7.

9.3.6. Оператор DD для выходного перфокарточного набора данных. Выходной набор данных на перфокартах имеет такую же организацию и такие же характеристики, что и входной.

Управляющий оператор DD для выходного перфокарточного набора данных может представляться в одной из следующих форм:

1) //*i* DD UNIT=a_1,DCB=(RECFM=a_2,BLKSIZE=a_3,
// LRECL=a_4)

2) //*i* DD SYSOUT=a_5

Здесь приняты следующие обозначения: a_1—код устройства вывода на перфокарты; a_2—формат блоков набора данных; a_3—длина блоков набора данных; a_4—длина физических записей; a_5 — класс устройства вывода на перфокарты (обычно B).

Форма 1 используется в том случае, если вывод данных нужно осуществлять через устройство, которое не определено как системное выходное устройство. В этом случае в качестве кода устройства обычно используется его физический адрес.

Форму 1 можно использовать для описания выходного перфокарточного набора данных с любым ссылочным номером. При использовании формата F длину физических записей можно не указывать. Если набор имеет стандартные характеристики (ссылочный номер 7, формат F, длину блоков 80 байтов), то параметр DCB можно опустить.

Форма 2 используется в том случае, если набор имеет формат F, длину блоков 80 байтов и вывод данных осуществляется через устройство вывода на перфокарты, которое в операционной системе определено как системное выходное устройство.

Форму 2 можно использовать для описания выходного перфокарточного набора с любым ссылочным номером и на практике она употребляется наиболее часто.

Вопросы и упражнения

1. Через какое устройство осуществляется вывод данных на перфокарты?

2. Назвать стандартные характеристики выходного перфокарточного набора данных.

3. Написать оператор DD для выходного перфокарточного набора данных, имеющего ссылочный номер 7, формат F и длину блока, равную 80 байтам. Вывод данных должен осуществляться:

а) через системное устройство вывода на перфокарты, имеющее класс B;

б) через устройство вывода на перфокарты, которое не определено как системное и имеет физический адрес 182.

4. Выполнить упражнение 3, предварительно заменив в нем ссылочный номер 7 на ссылочный номер 4.

5. Выполнить упражнение 3, предварительно заменив в нем ссылочный номер 7 на ссылочный номер 5.

9.3.7. Оператор DD для выходного набора данных на бумажной ленте. Набор данных на бумажной ленте представляет собой последовательность строк символов. Он относится к наборам последовательного доступа, может иметь блоки с форматами UA,FA,FBA, VA,VBA (при работе с управляющими символами) и с форматами U,F,FB,V,VB (при работе без управляющих символов). Длины блоков и физических записей этого набора при использовании форматов U,F,FB,V,VB не должны превышать 128 символов (длины строки бумажной ленты), а при использовании форматов UA,FA, FBA,VA,VBA — 129 символов.

Обращение к набору может осуществляться только через операторы форматного вывода.

Устройство построчной печати (АЦПУ), осуществляющее выдачу данных на бумажную ленту, не обеспечивает возврата ни на начало набора, ни на предыдущую запись, поэтому при работе с таким устройством операторы BACKSPACE и REWIND не вызывают никаких действий.

Управляющий оператор DD для выходного набора данных на бумажной ленте может представляться в одной из следующих форм:

1) //*i* DD UNIT=a_1,DCB=(RECFM=a_2,
   // BLKSIZE=a_3,LRECL=a_4)
2) //*i* DD SYSOUT=a_5

Здесь приняты следующие обозначения: a_1 — код устройства; a_2 — формат блоков набора данных; a_3 — длина блока в символах; a_4 — длина физической записи в символах; a_5 — класс АЦПУ (обычно A).

Форма 1 используется в том случае, если вывод данных нужно осуществлять через устройство, которое не определено как системное выходное устройство. В этом случае в качестве кода устройства обычно используется его физический адрес. Форму 1 можно использовать для описания выходного набора данных с любым ссылочным номером.

Если набор имеет стандартные характеристики (ссылочный номер 6, формат блоков UA, максимальная длина блоков 129 символов), то параметр DCB можно опустить.

Форма 2 используется в том случае, если вывод данных осуществляется через устройство, которое в операционной системе определено как системное выходное устройство. Ее можно использовать для описания наборов с любым ссылочным номером, но с форматом блоков UA и с максимальной длиной физических записей 129 символов.

Вопросы и упражнения

1. Через какое устройство осуществляется вывод данных на бумажную ленту?

2. Назвать стандартные характеристики выходного набора данных на бумажной ленте.

3. Написать оператор DD для выходного набора данных на бумажной ленте, имеющего ссылочный номер 6, формат UA и максимальную длину блока, равную 129 байтам. Вывод данных должен осуществляться:

а) через системное АЦПУ, имеющее класс А;

б) через АЦПУ, которое не определено как системное и имеет физический адрес 182.

4. Выполнить упражнение 3, предварительно заменив в нем ссылочный номер 6 на ссылочный номер 5.

5. Выполнить упражнение 3, предварительно заменив в нем формат UA на формат U и длину блока, равную 129 байтам, на длину блока, равную 128 байтам.

9.3.8. Оператор DD для входных и выходных наборов данных, размещаемых на магнитных дисках. Магнитные диски относятся к устройствам прямого доступа и могут использоваться для размещения на них наборов данных всех типов (прямого доступа, последовательного доступа, входные, выходные и т. д.).

Данные в них могут храниться в машинной и в символьной формах, блоки могут иметь любой из предусмотренных в операционной системе форматов.

Отдельный пакет дисков в операционной системе ОС ЕС ЭВМ принято называть томом. Нами будут рассмотрены описания лишь однотомных наборов данных (размещаемых в пределах одного тома), длина блока которых не превосходит длины дорожки пакета дисков. Работа с наборами данных на дисках осуществляется только в режиме со стандартными метками.

Каждый том имеет стандартную метку тома, т. е. специальную область, в которой хранятся общие сведения о томе: так называемый серийный номер тома и адрес оглавления тома. Оглавление тома — это специальный набор данных, в котором хранятся стандартные метки всех наборов данных, размещенных в данном томе. Каждая метка набора данных занимает в оглавлении отдельную запись. Количество меток для каждого набора зависит от типа этого набора и от способа размещения набора в томе.

В метке наборов помещаются такие сведения, как имена наборов данных, серийные номера томов, даты создания наборов, физические

характеристики наборов (форматы блоков, размеры блоков и физических записей), адреса наборов и т. д.

Метка тома и оглавление тома создаются обслуживающим персоналом ЭВМ в процессе подготовки тома к работе (в процессе инициализации тома) по специальной обслуживающей программе. Метки наборов данных помещаются в оглавление тома в процессе создания этих наборов (на этапе открытия наборов данных). Необходимые для меток сведения берутся из управляющих операторов задания (из операторов DD). При работе с ранее созданными наборами физические характеристики наборов в операторах DD указывать необязательно, так как программы ввода-вывода будут брать эти характеристики из метки набора.

Метки томов и наборов данных используются операционной системой для контроля правильности установки томов на дисководы, для поиска наборов данных в томе и для других целей. Допускается одновременная работа с любым количеством наборов данных, размещенных в одном томе. Оператор DD для наборов данных, размещаемых на магнитных дисках, имеет вид:

```
//t DD DSNAME=a1,VOLUME=a2,UNIT=a3,
//       DISP=a4,SPACE=a5,DCB=a6
```

где t — имя оператора DD; DSNAME, VOLUME, DISP и т. д.— имена ключевых параметров; a_1, a_2, . . ., a_6 — значения ключевых параметров.

Ниже приводятся описания этих параметров.

1) Параметр DSNAME.

Этот параметр имеет вид:

```
DSNAME=a
```

где a — либо имя набора данных, либо ссылка на оператор DD, содержащий нужное имя набора данных.

Имя может быть простым и составным. Простое имя — это идентификатор, содержащий до восьми символов. Составное имя состоит из нескольких простых, отделенных друг от друга точками. Число символов в составном имени не должно превышать 44. Приведем несколько примеров имен наборов данных:

```
MASSIV
B1.MASSIV
DATA1.B1.REZ3
```

Операционная система предусматривает возможность использования временных наборов данных. Эти наборы создаются в про-

цессе выполнения шага задания и могут использоваться в других шагах этого задания. Временным наборам могут присваиваться только простые имена, которые должны начинаться с символа &, являющегося признаком временных наборов. Временные наборы, созданные в процессе выполнения задания, после завершения задания уничтожаются.

Если в параметре DSNAME нужно указать имя набора, который был уже описан в данном задании, то это можно сделать путем указания ссылки на оператор DD, содержащий это имя.

Ссылка на оператор DD имеет вид:

$$*.t_1.t_2$$

где t_1 — имя шага задания (оператора EXEC), в состав которого входит оператор DD; t_2 — имя оператора DD, содержащего нужное нам имя набора. Если ссылка осуществляется на оператор DD, содержащийся в том же шаге, то имя шага задания не указывается. В этом случае ссылка будет иметь вид:

$$*.t_2$$

Параметр DSNAME может быть опущен только для вновь создаваемого набора. В этом случае набор считается временным, и ему дается стандартное имя. При использовании этого набора в других шагах задания его имя можно указать только путем ссылки на тот оператор DD, в котором этот набор был впервые описан.

Имя параметра DSNAME можно записывать в сокращенном виде: DSN. Приведем несколько примеров записи параметра DSNAME:

а) DSN=REZ1
б) DSNAME=M1.DISK
в) DSN=&M2
г) DSN=*.L1.FT02F001
д) DSNAME=*.FT15F001

2) Параметр VOLUME.

В рассматриваемом нами упрощенном варианте языка управления заданиями параметр VOLUME содержит один ключевой подпараметр SER и имеет вид:

$$\text{VOLUME}=\text{SER}=a$$

где a — серийный номер тома, на котором размещается описываемый набор данных. Серийный номер тома может содержать от одного до шести символов, причем в качестве таких символов могут исполь-

зоваться только цифры и буквы латинского алфавита. Имя параметра может записываться в сокращенном виде: VOL.

Серийный номер тома может не указываться только для вновь создаваемых наборов данных. В этом случае система сама находит ему место на одном из тех томов, которые установлены на устройства ввода-вывода. Серийный номер выбранного тома указывается в сообщении о распределении устройств под наборы данных.

Примеры записей параметра VOLUME:

а) VOLUME=SER=111222

б) VOL=SER=FOR15

3) Параметр UNIT.

В рассматриваемом нами варианте языка управления заданиями параметр UNIT содержит один позиционный параметр и имеет вид:

UNIT=*a*

где *a*—шифр устройства (физический адрес, типовое или групповое имя), на которое должен быть помещен том, содержащий описываемый набор данных.

Примеры записей параметра UNIT:

а) UNIT=132 (код устройства задан физическим адресом);

б) UNIT=5050 (код устройства задан типовым именем);

в) UNIT=SYSDA (код устройства задан групповым именем).

Параметр UNIT должен присутствовать всегда за исключением случаев, описанных в пункте 4.

4) Параметр DISP.

Этот параметр служит для указания состояния набора данных и для указания того, что система должна делать с набором после нормального завершения пункта задания и после аварийного его завершения.

Параметр DISP имеет вид:

DISP=(a_1,a_2,a_3)

Подпараметр a_1 определяет состояние набора и может принимать одно из следующих значений: OLD, SHR, NEW, MOD.

Состояние NEW (новый) указывается для наборов, которые создаются на данном шаге задания.

Состояние OLD (старый) указывается для наборов, которые были созданы либо в предыдущих шагах данного задания, либо в других заданиях. Состояние OLD говорит о том, что данный набор отдается шагу задания в монопольное использование. Это означает,

что все запросы на этот набор из других заданий исполняться не будут до окончания данного шага задания.

Состояние SHR (разделяемый) указывается для ранее созданных наборов данных, которые могут одновременно использоваться несколькими заданиями. Обычно такое состояние приписывается наборам, которые используются как входные.

Состояние MOD (модифицируемый) указывается для ранее созданных наборов последовательного доступа, в которые будут добавляться записи. Добавляемые записи размещаются следом за последней, ранее созданной записью.

Подпараметр a_2 указывает операционной системе на то, что нужно сделать с набором данных после нормального завершения шага задания. Он может принимать следующие значения: KEEP, DELETE, PASS, CATLG, UNCATLG.

Значение KEEP означает, что набор данных надо сохранить.

Значение DELETE означает, что набор данных необходимо уничтожить.

Значение PASS означает, что набор данных передается последующим шагам задания. В этом случае информация об устройстве и томе запоминается операционной системой. Поэтому при описании такого набора в последующих шагах задания параметры UNIT и VOLUME описывать необязательно.

Значение CATLG означает, что набор данных после окончания шага задания должен быть каталогизирован. В этом случае имя набора данных, а также информация о томе и об устройстве заносятся в специальный системный каталог наборов данных. Если в задании используется ранее каталогизированный набор данных, то оператор DD, описывающий этот набор, должен содержать только два параметра: DSNAME и DISP.

Значение UNCATLG означает, что набор данных после окончания шага задания удаляется из каталога.

Если параметр a_2 не был задан, то ему приписывается стандартное значение. Для временных и новых наборов стандартным значением подпараметра a_2 является DELETE, а для старых и разделяемых — KEEP.

Значение подпараметра a_3 указывает системе, что нужно сделать с набором данных в случае аварийного завершения шага задания. Он может принимать те же значения, что и подпараметр a_2, кроме значения PASS. Временный набор данных в случае аварийного завершения шага задания всегда уничтожается.

Если a_3 не указан, то при аварийном завершении шага задания выполняются действия, заданные параметром a_2.

Примеры записей параметра DISP:

```
DISP=(NEW,KEEP)
DISP =(NEW,PASS,DELETE)
DISP=OLD
```

Первая запись означает, что набор создается во время выполнения шага задания, и он должен быть сохранен как после успешного завершения шага, так и после аварийного его завершения.

Вторая запись означает, что набор данных создается во время выполнения шага задания. При нормальном завершении шага задания его надо передать последующим шагам, а при аварийном — удалить из тома (уничтожить).

Третья запись означает, что в программе будет использован ранее созданный набор. После завершения программы (успешного или аварийного) набор должен быть сохранен.

5) Параметр SPACE.

Этот параметр указывается только для вновь создаваемых наборов. С его помощью задается объем памяти на диске, необходимый для размещения набора данных. Он записывается следующим образом:

$$SPACE=(a_1, (a_2, a_3), a_4)$$

Подпараметр a_1 определяет единицу измерения объема памяти. Он может принимать следующие значения:

TRK (если память указывается в дорожках);

CYL(если память указывается в цилиндрах);

целое число без знака, означающее размер блока памяти в байтах (если память измеряется в блоках).

Второй подпараметр (a_2) представляется в виде целого числа без знака, означающего первоначальное количество единиц памяти, выделяемое под данный набор.

Третий подпараметр (a_3) означает количество единиц памяти, которое дополнительно будет выделяться под набор данных, если первоначальное количество окажется недостаточным. Дополнительный объем памяти может выделяться набору до 15-ти раз. Четвертый подпараметр (a_4) может принимать только одно значение в виде слова RLSE, которое указывает, что после завершения создания набора данных выделенная ему, но не использованная память освобождается.

Третий и четвертый подпараметры могут отсутствовать. В этих случаях параметр SPACE будет представляться в следующих формах:

а) $SPACE=(a_1,(a_2,a_3))$ — отсутствует четвертый подпараметр;

б) SPACE=(a_1,a_2,a_4) — отсутствует третий подпараметр;

в) SPACE=(a_1,a_2) — отсутствует третий и четвертый подпараметры.

При создании библиотечных наборов данных параметр SPACE должен содержать еще один подпараметр, который указывает сколько блоков длиной по 256 байтов отводится под оглавление библиотеки. В каждый из таких блоков помещаются сведения в среднем о пяти разделах библиотечного набора. Дополнительный подпараметр записывается в виде целого числа без знака следом за подпараметром a_3. В этом случае параметр SPACE будет выглядеть так:

SPACE=(a_1,(a_2,a_3,b),a_4)

где b— дополнительный подпараметр.

Рассмотрим несколько примеров записи параметра SPACE.

Пример 1. SPACE=(TRK,10). Под набор данных будет выделено 10 дорожек.

Пример 2. SPACE=(80,(500,100)). Первоначально под набор данных будет выделено 500 блоков по 80 байтов. Если в процессе создания набора этого объема памяти окажется недостаточно, то под него будет выделено дополнительно 100 таких блоков. Если и этого объема памяти окажется недостаточно, то под него будет выделено еще 100 блоков, и т. д. Такое выделение может осуществляться до 15-ти раз.

Пример 3. SPACE=(CYL,25,RLSE). Под набор данных будет выделено 25 цилиндров. Если в процессе создания набора данных он займет 15 цилиндров, то после выполнения шага задания десять цилиндров будут у набора отобраны.

6) Параметр DCB.

Если набор данных последовательного доступа имеет характеристики, отличные от стандартных, то их нужно указать в параметре DCB, который в общем случае имеет вид:

DCB=(RECFM=a_1,BLKSIZE=a_2,LRECL=a_3,BUFNO=a_4)

Подпараметр RECFM определяет формат блоков набора данных и может принимать следующие значения: F, FB, V, VB, VS, VBS, U.

В качестве значения подпараметра BLKSIZE указывается число, равное длине или максимальной длине блока в байтах (символах).

В качество значения подпараметра LRECL указывается число, равное длине или максимальной длине физической записи в байтах (символах).

Между числами a_2 и a_3 должны выдерживаться определенные соотношения, которые зависят от значения подпараметра RECFM:

$a_2=a_3$, если RECFM=F или RECFM=U;

$a_2 = n \cdot a_3$ (n целое число), если RECFM=FB;

$a_2=n \cdot a_3+4$ (n целое число), если RECFM принимает одно из следующих значений: V, VB.

Для наборов данных с форматами F и U подпараметр LRECL можно опускать.

В качестве значения подпараметра BUFNO указываются числа 1 или 2, которые определяют количество буферов в оперативной памяти, выделяемое данному набору. Если этот параметр не указан, то будет выделено 2 буфера. Приведем несколько примеров записей параметра DCB:

а) DCB=BUFNO=1

б) DCB=(RECFM=FB,LRECL=80,BLKSIZE=400)

в) DCB=(RECFM=VB,LRECL=50,BLKSIZE=204, BUFNO=1)

Если в параметре DCB нужно указать характеристики, совпадающие с характеристиками набора, уже описанного в данном задании, то это можно сделать путем указания ссылки на оператор DD, содержащий эти характеристики. Ссылка указывается в качестве значения параметра DCB. Например, запись DCB=*.D1.FT02F001 означает, что данному набору будут приписаны такие же характеристики, что и набору FT02F001, описанному на шаге D1.

Запись DCB=*.FT08F001 означает, что данному набору будут приписаны такие же характеристики, что и набору FT08F001, описанному в данном шаге.

Для наборов данных прямого доступа параметр DCB можно не указывать, так как необходимые характеристики набора данных содержатся в программе (в операторе DEFINE FILE).

Вопросы и упражнения

1. Пусть оператор DD для набора данных последовательного доступа, размещаемого на магнитном диске, имеет вид:

```
//FT01F001  DD  DSN=REZ,VOL=SER=D00015,
//                UNIT=5050,DISP=(NEW,KEEP),
//                SPACE=(TRK,20),DCB=(LRECL=80,
//                BLKSIZE=84,RECFM=V)
```

Ответить на следующие вопросы:

а) Какая последовательность символов запишется в метку набора в качестве его имени?

б) Сколько места займет набор данных на магнитном диске?

в) Какой серийный номер имеет том, на котором размещается набор данных?

г) Что произойдет с набором данных после выполнения задания?

д) Каков формат у блоков набора данных?

е) Какова длина (или максимальная длина) блоков и физических записей этого набора?

ж) Каков ссылочный номер набора данных?

2. Пусть некоторая программа создает набор данных последовательного доступа со ссылочным номером 3. Набор имеет формат F, длина блоков равна 128 байтам. Набор нужно разместить на пакете магнитных дисков с серийным номером тома 221223, выделив ему 15 цилиндров. Дисковод, на который будет установлен том, имеет физический адрес 023.

Назначить имя этому набору и написать для него управляющий оператор DD.

3. Пусть задание на выполнение программ содержит два шага. Первый шаг содержит управляющие операторы:

```
//STEP1 EXEC PGM=PROG1
//FT01F001  DD  UNIT=5050,VOL=SER=D15001,
//              DISP=(NEW,PASS),SPACE=(TRK,40)
```

На этом шаге создается временный набор данных со стандартным именем, который передается второму шагу задания.

На втором шаге задания выполняется программа PROG2, которая находится в той же библиотеке, что и программа PROG1. Эта программа использует в своей работе набор данных, созданный на первом шаге, но со ссылочным номером 2. После ее выполнения этот набор подлежит уничтожению.

Написать для второго шага задания: 1) оператор EXEC, вызывающий на исполнение программу PROG2; 2) оператор DD, описывающий набор данных, созданный на первом шаге, и используемый в программе PROG2.

9.3.9. Оператор DD для входных и выходных наборов данных, размещаемых на магнитных лентах. Магнитные ленты относятся к устройствам последовательного доступа и могут использоваться для размещения в них входных и выходных наборов данных последовательного доступа. Данные в этих наборах могут храниться в машинной и символьной формах, блоки могут иметь любой из предусмотренных форматов, длина блока может колебаться от 18 до 32760 байтов.

Отдельную бобину ленты в ОС ЕС ЭВМ принято называть томом. Нами будут рассмотрены описания лишь однотомных наборов, каждый из которых может занимать либо часть тома, либо целый том.

Каждый том имеет специальные маркеры начала и конца тома. Наборы данных начинают размещаться от маркера начала тома и отделяются друг от друга специальными блоками, называемыми лепточными марками. Марки записываются на том автоматически, как признаки конца наборов (в моменты закрытия создаваемых наборов).

Программы на языке фортран могут использовать два режима работы с магнитными лентами: со стандартными метками и без меток.

Набор данных без меток — это совокупность блоков данных, расположенных между двумя ленточными марками или же между маркером начала тома и ленточной маркой.

При использовании наборов данных без меток операционная система осуществляет подвод нужного набора, используя информацию, содержащуюся в управляющих операторах задания, однако она не контролирует ни правильность установки тома на соответствующий магнитофон, ни правильность подвода нужного набора. Этот режим работы используется редко. Более надежным режимом является режим со стандартными метками.

Стандартная метка — это специальный блок длиною 80 байтов. Набор данных со стандартными метками имеет следующую структуру: в начале набора располагается группа меток начала набора, затем идут блоки данных, а затем группа меток конца набора. Обычно каждая из групп меток содержит по 2 стандартные метки набора. Метки записываются в набор автоматически во время его создания (метки начала набора — в момент его открытия, метки конца набора — в момент закрытия).

Том магнитной ленты, предназначенный для размещения наборов данных со стандартными метками, должен быть предварительно подготовлен (инициализирован) с помощью специальной обслуживающей программы. Это осуществляется обслуживающим персоналом ЭВМ. Инициализация заключается в создании на магнитной ленте стандартной метки тома, которая размещается после маркера начала. В метку тома помещается серийный номер тома, который берется из управляющих карт (из оператора описания набора данных).

Серийный номер тома может содержать от одного до шести буквенно-цифровых символов и используется операционной системой для контроля правильности установки на магнитофон нужного тома.

В первую метку начала набора данных помещаются такие сведения, как имя набора данных, серийный номер тома, порядковый

номер набора в томе, дата создания набора и некоторая другая информация.

Во вторую метку начала набора помещаются физические характеристики набора данных: форматы блоков, размеры блоков и физических записей, плотность записи. Все перечисленные выше сведения помещаются в метки в процессе создания набора и берутся из управляющих операторов задания. При работе с уже созданным набором его физические характеристики указывать в операторе описания набора не обязательно. Для обеспечения работы системы ввода-вывода эти характеристики будут браться из метки.

В первую и вторую метки конца набора помещаются такие же сведения, как и в первую и вторую метки начала тома.

Одновременная работа с наборами, размещенными в одном томе, не разрешена. Это означает, что в одном томе может находиться в открытом состоянии только один набор данных. Попытка открыть второй набор без предварительного закрытия первого приводит к аварийному завершению задачи.

Первоначальное создание наборов, расположенных в одном томе магнитной ленты, должно вестись в том порядке, в котором они располагаются в томе. После создания одного набора он закрывается, после чего создается следующий набор. Порядок расположения наборов в томе указывается в управляющих операторах задания.

При чтении созданных наборов они могут перебираться в любом порядке. Если засылка данных осуществляется в набор, который был создан ранее, то все последующие записи этого набора и все последующие наборы этого тома становятся недоступными для дальнейшего использования.

Оператор DD для входных и выходных наборов данных, размещенных на магнитных лентах, имеет вид:

```
//i  DD DSNAME=a1,VOLUME=a2,UNIT=a3,
//        DISP=a4,LABEL=a5,DCB=a6
```

где i — имя оператора DD; DSNAME, VOLUME и т. д.— имена ключевых параметров; $a_1, a_2, \ldots, a_6$— значения ключевых параметров.

Описания параметров DSNAME,VOLUME,UNIT,DISP,DCB приведены в разделе 9.3.8 (оператор DD для входных и выходных наборов данных, размещенных на магнитных дисках). Ниже приводится описание параметра LABEL.

Параметр LABEL в рассматриваемом нами варианте языка управления заданиями имеет вид:

LABEL=(a_1,a_2)

где a_1 — целое число, означающее порядковый номер набора данных на магнитной ленте; a_2 — указатель меток набора данных.

Подпараметр a_2 может принимать два значения: NL — когда метки отсутствуют, SL — когда используются стандартные метки.

Если подпараметр a_1 опущен, то он принимается равным единице, а если подпараметр a_2 опущен, то он принимается равным SL. Отсутствие параметра LABEL говорит о том, что набор данных имеет стандартные метки и является первым в томе.

Пример записей параметра LABEL:

а) LABEL=(3,NL)

б) LABEL=(5)

в) LABEL=(,NL)

Вопросы и упражнения

1. Пусть последовательный набор данных описан в операторе DD следующим образом:

```
//FT28F001 DD DSN=*.FT08F001,VOL=SER=ABCD,
//   LABEL=3,UNIT=SYSSQ,DISP=OLD
```

Ответить на следующие вопросы:

а) Какой ссылочный номер имеет данный набор данных в программе?

б) Где (на каком шаге задания и в каком операторе) определено имя набора данных)?

в) На каком устройстве размещается набор данных?

г) Какой серийный номер имеет том, на котором размещен данный набор?

д) Какой режим работы используется с указанным в операторе томом?

е) Какой порядковый номер имеет данный набор в томе?

ж) Будут ли удовлетворяться запросы на этот набор из других заданий во время выполнения данного шага задания?

2. Пусть в одном шаге задания описаны следующие два оператора DD:

```
//FT18F001  DD  DSN=MAC18,DISP=SHR,UNIT=5010,
//   VOL=SER=111222,LABEL=(2,NL)
//FT01F001 DD DDNAME=FT18F001
```

Написать, какие характеристики имеет набор данных со ссылочным номером 1.

3. Пусть оператор DD для последовательного набора данных со ссылочным номером 2 имеет вид:

```
//  FT02F001 DD DSN=&NOM,DISP=NEW,UNIT=SYSSQ,
//  SPACE=(800,200)
```

Ответить на следующие вопросы:

а) На каких устройствах может быть размещен данный набор?

б) Можно ли к данному набору обращаться на следующем шаге задания?

в) Какой формат записи и размер блоков будет иметь набор со ссылочным номером 2?

4. Пусть в программе вводятся данные из набора со ссылочным номером 38, который размещен на магнитной ленте и имеет следующие характеристики:

а) серийный номер тома L00011;

б) имя набора TXP;

в) набор данных снабжен стандартными метками;

г) набор является вторым на данном томе.

Описать данный набор в операторе DD.

9.3.10. Оператор DD для библиотеки задания и библиотеки шага задания. Оператор DD, описывающий библиотеку задания, должен иметь вид:

```
//JOBLIB DD DSN=a1,UNIT=a2,VOL=SER=a3,
//  DISP=SHR
```

Оператор DD, описывающий библиотеку шага задания, должен иметь вид:

```
//STEPLIB DD DSNAME=a1,UNIT=a2,VOL=SER=a3,
//  DISP=SHR
```

Здесь приняты следующие обозначения: a_1 — имя библиотечного набора данных, в котором хранятся загрузочные модули библиотеки; a_2 — код устройства, на которое устанавливается том (пакет магнитных дисков), содержащий библиотеку; a_3 — серийный номер тома, содержащего библиотеку.

Так, например, оператор

```
//JOBLIB DD DSN=BIBL1,UNIT=5050,
//  VOL=SER=D24,DISP=OLD
```

описывает библиотеку задания с именем BIBL1, которая размещается на пакете магнитных дисков типа 5050 с серийным номером D24

9.3.11. Цепочки наборов данных. Если несколько наборов данных последовательного или библиотечного методов доступа имеют

одинаковые характеристики (методы доступа, форматы блоков, размеры блоков и физических записей), то их можно описать в управляющем операторе DD как один набор. В этом случае оператор DD должен иметь вид:

```
//t  DD p1
//   DD p2
     . . .
//   DD pn
```

где p_1 — совокупность параметров оператора DD, относящегося к 1-му набору; p_2 — совокупность параметров оператора DD, относящегося ко второму набору и т. д. Такие объединенные наборы данных могут использоваться только как входные (только для чтения данных).

Если наборы имеют последовательную организацию, то чтение данных осуществляется сначала из первого набора, затем из второго и т. д. Если наборы имеют библиотечную организацию и используются в качестве библиотеки задания или шага задания, то поиск программы будет осуществляться сначала в первом наборе, затем во втором и т. д. Так, например, описание библиотеки задания может иметь вид:

```
//JOBLIB DD DSN=BIBL1, DISP=SHR
//          DD DSN=BIBL2, DISP=SHR
```

где BIBL1 и BIBL2 — каталогизированные библиотечные наборы, которые содержат нужные нам программы (загрузочные модули).

Для программ, написанных на языке ассемблера, в состав объединенного набора данных иногда могут включаться наборы данных с различными характеристиками.

9.3.12. Примеры на составление заданий. П р и м е р 1.

Пусть для решения некоторой задачи требуется выполнить программы A, B и C, которые содержатся в библиотеке загрузочных модулей BIBL. Библиотека находится на пакете магнитных дисков типа 5050 с серийным номером D01. Программа A, выполняемая первой, осуществляет ввод исходных данных с перфокарт и создает на магнитном диске типа 5061 набор данных последовательного доступа, который используется программами B и C. Характеристики перфокарточного набора: ссылочный номер набора 5, формат блоков F, длина блоков 80 байтов. Характеристики набора на магнитном диске: ссылочный номер набора 8, формат блоков VB, максимальная длина блоков 112 байтов, максимальная длина физической записи 54 байта, требуемый объем памяти 150 дорожек. Программа B

выполняется второй, осуществляет ввод исходных данных с перфокарт и добавляет в набор данных, созданный программой A на магнитном диске, ряд блоков. Характеристики перфокарточного набора: ссылочный номер набора 5, формат блоков F, длина блоков 80 байтов. Набор на магнитном диске используется в программе B со ссылочным номером 9. Программа C использует набор на магнитном диске, созданный программами A и B, как входной (для ввода исходных данных), а свои результаты выводит на бумажную ленту через АЦПУ.

Набор на бумажной ленте используется в программе C со ссылочным номером 6, а набор на магнитном диске — со ссылочным номером 8. После выполнения программы набор 8 подлежит уничтожению. Каждая из программ требует для своего исполнения 50 К байтов оперативной памяти.

Задание на решение описанной задачи может выглядеть так:

```
//N1 JOB CLASS=B,REGION=50K
//JOBLIB DD  DSN=BIBL,DISP=SHR,UNIT=5050,
//              VOL=SER=D01
//  EXEC PGM=A
//FT05F001 DD *
<Исходные данные программы A>
/*
//FT08F001 DD  DSN=&ND8,DISP=(NEW,PASS),
//              UNIT=5061,SPACE=(TRK,150),
//              DCB=(RECFM=VB,BLKSIZE=112,
//              LRECL=54)
//  EXEC PGM=B
//FT05F001 DD *
<Исходные данные программы B>
/*
//FT09F001 DD DSN= &ND8,  DISP=(MOD,PASS)
//  EXEC PGM=C
//FT08F001 DD DSN=&ND8,DISP=(OLD, DELETE)
//FT06F001 DD SYSOUT=A
//
```

Так как набор на магнитном диске является временным, то имя ему можно не давать. В этом случае задание на решение задачи можно представить в таком виде:

```
//N1 JOB CLASS=B, REGION=50K
//JOBLIB DD DSN=BIBL, DISP=SHR,
```

```
//  UNIT=5050, VOL=SER=D01
//ST1 EXEC PGM=A
//FT05F001 DD *
```
<Исходные данные программы A>
```
/*
//FT08F001  DD DISP=(NEW,PASS),UNIT=5061,
//             SPACE=(TRK,150),DCB=(RECFM=VB,
//             BLKSIZE=112,LRECL=54)
// EXEC PGM=B
//FT05F001 DD *
```
<Исходные данные программы B>
```
/*
//FT09F001 DD DSN=*.ST1.FT08F001,
//             DISP=(MOD,PASS)
//  EXEC PGM=C
//FT08F001 DD DSN=*.ST1.FT08F001,
//             DISP=(OLD,DELETE)
//FT06F001 DD SYSOUT=A
//
```

Пример 2. Пусть требуется решить задачу, описанную в примере 1, но программы A, B и C требуется запускать по отдельным заданиям. В этом случае набор данных, создаваемый программой A на магнитном диске, должен передаваться от задания к заданию и не может быть временным. Ниже приводятся 2 варианта заданий на запуск этих программ.

Вариант 1 (набор данных не каталогизируется).

Задание на запуск программы A:

```
//PROGA1 JOB CLASS=B,REGION=50K
//JOBLIB DD DSN=BIBL,DISP=SHR,
//          UNIT=5050,VOL=SER=D01
// EXEC PGM=A
//FT05F001 DD *
```
<Исходные данные программы A>
```
/*
//FT08F001 DD DSN=ND8,DISP=(NEW,KEEP),
//             UNIT=5061,VOL=SER=S1S1,
//             SPACE=(TRK,150),
//             DCB=(RECFM=VB,BLKSIZE=112,
//             LRECL=54)
//
```

Задание на запуск программы B:

```
//PROGB1 JOB CLASS=B,REGION=50K
//JOBLIB DD DSN=BIBL,DISP=SHR,
//          UNIT=5050,VOL=SER=D01
//  EXEC PGM=B
//FT05F001 DD *
   (Исходные данные программы B)
/*
//FT09F001 DD DSN=ND8,DISP=MOD,
//            UNIT=5061,VOL=SER=S1S1
//
```

Задание на запуск программы C:

```
//PROGC1 JOB CLASS=B,REGION=50K
//JOBLIB DD DSN=BIBL,DISP=SHR,
//          UNIT=5050,VOL=SER=D01
//  EXEC PGM=C
//FT08F001 DD DSN=ND8,DISP=(OLD,DELETE),
//            UNIT=5061,VOL=SER=S1S1
//FT06F001 DD SYSOUT=A
```

Вариант 2 (набор данных, создаваемый на магнитном диске, каталогизируется).

Задание на запуск программы A:

```
//PROGA2 JOB CLASS=B,REGION=50K
//JOBLIB DD DSN=BIBL,DISP=SHR,
//          UNIT=5050,VOL=SER=D01
// EXEC PGM=A
//FT05F001 DD *
   (Исходные данные программы A)
/*
//FT08F001 DD DSN=ND8,DISP=(NEW,CATLG),
//            UNIT=5061,SPACE=(TRK,150),
//            DCB=(RECFM=VB,BLKSIZE=112,
//            LRECL=54)
```

Задание на запуск программы B:

```
//PROGB2 JOB CLASS=B,REGION=50K
//JOBLIB DD DSN=BIBL,DISP=SHR,
//          UNIT=5050,VOL=SER=D01
//  EXEC PGM=B
//FT05F001 DD *
```

```
    (Исходные данные программы B)
/*
//FT09F001 DD DSN=ND8,DISP=MOD
//
```

Задание на запуск программы C:

```
//PROGC2 JOB CLASS=B,REGION=50K
//JOBLIB DD DSN=BIBL,DISP=SHR,
//             UNIT=5050,VOL=SER=D01
//  EXEC PGM=C
//FT08F001 DD DSN=ND8,DISP=(OLD,DELETE)
//FT06F001 DD  SYSOUT=A
//
```

Глава 10

ТРАНСЛЯЦИЯ И РЕДАКТИРОВАНИЕ ПРОГРАММ

10.1. Каталогизированные процедуры

В процессе подготовки программ к выполнению на ЭВМ программисту приходится составлять задания на выполнение таких системных программ, как транслятор, редактор связей и др. Для составления заданий необходимо знать такие характеристики исполняемых программ, как их имена, требуемые объемы оперативной памяти, перечни наборов данных, используемых в программах, размеры этих наборов, типы устройств, на которых можно размещать эти наборы, форматы и размеры блоков и т. д. Это приводит к тому, что составление заданий на выполнение системных программ выливается в кропотливую и трудоемкую работу.

В операционной системе ОС ЕС ЭВМ имеются средства, позволяющие существенно облегчить работу по составлению заданий на исполнение часто используемых программ, в том числе транслятора и редактора связей. К таким средствам относится аппарат каталогизированных процедур.

Каталогизированная процедура представляет собой фрагмент задания, содержащий один или несколько шагов и включающий в себя операторы шага задания (EXEC), через которые осуществляется вызов программ на исполнение, и операторы описания наборов данных (DD), используемых в этих программах. Каталогизированным процедурам приписываются имена (идентификаторы), и они помещаются в системную библиотеку каталогизированных процедур с именем SYS1.PROCLIB.

Вызов каталогизированной процедуры осуществляется оператором шага задания, который должен представляться в одной из следующих форм:

а) //*t* EXEC PROC=*a*

б) //*t* EXEC *a*

где *t* — имя оператора EXEC, *a* — имя каталогизированной процедуры, текст которой содержится в системной библиотеке каталогизированных процедур.

Вызов каталогизированной процедуры осуществляется операционной системой в процессе выполнения задания, в результате чего оператор вызова каталогизированной процедуры будет заменен на текст каталогизированной процедуры, взятый из библиотеки.

Пример 1. Пусть заданы:

а) каталогизированная процедура с именем P1, содержащая операторы

```
//ST1 EXEC PGM=A1,PARM=(5,14)
//N1 DD DSN=NAB1,DISP=OLD,
//       UNIT=5050,VOL=SER=D17
//ST2 EXEC PGM=A2
//N2 DD SYSOUT=A
```

б) задание, содержащее операторы

```
//S1 JOB  REGION=80K
//ST EXEC PROC=P1
//
```

После вызова каталогизированной процедуры задание примет вид:

```
//S1 JOB REGION=80K
//ST1 EXEC PGM=A1,PARM=(5,14)
//N1 DD DSN=NAB1,DISP=OLD,
//       VOL=SER=D17,UNIT=5050
//ST2 EXEC PGM=A2
//N2 DD SYSOUT=A
//
```

В каталогизированную процедуру запрещается включать: оператор DD, описывающий библиотеку задания (JOBLIB DD...); операторы DD с параметрами DATA и *; оператор вызова другой процедуры.

При вызове каталогизированной процедуры можно изменять в ней параметры операторов EXEC и DD, а также включать в нее дополнительные операторы DD.

Информация об изменении параметров операторов EXEC помещается в оператор вызова каталогизированной процедуры,

который должен представляться в одной из следующих форм:

а) //*i* EXEC PROC=*a*, $p_1.j_1=a_1, p_2.j_2=a_2, \ldots, p_n.j_n=a_n$

б) //*i* EXEC *a*, $p_1.j_1=a_1, p_2.j_2=a_2, \ldots, p_n.j_n=a_n$

где *i* — имя оператора вызова каталогизированной процедуры; *a* — имя каталогизированной процедуры; p_k ($k = 1, 2, \ldots, n$) — имя изменяемого параметра; j_k — имя оператора EXEC, к которому относится параметр p_k; a_k — либо пусто, либо новое значение параметра p_k.

Если оператор EXEC с именем j_k, входящий в состав каталогизированной процедуры *a*, содержит параметр с именем p_k, то он либо удаляется из оператора EXEC (когда a_k пусто), либо его значение заменяется на a_k (когда a_k не пусто). Если же не содержит, то в состав оператора EXEC добавляется параметр $p_k = a_k$.

Новое значение изменяемого параметра нужно записывать полностью (со всеми необходимыми подпараметрами), даже если некоторые из его подпараметров остаются в прежнем виде.

Если требуется изменить параметры в нескольких операторах EXEC каталогизированной процедуры, то в операторе вызова каталогизированной процедуры вначале записываются все изменения, относящиеся к первому (по порядку следования) изменяемому оператору EXEC, затем ко второму и т. д.

Пример 2. Пусть заданы:

а) каталогизированная процедура с именем P2, содержащая операторы

```
//ST1 EXEC PGM=PROG1,PARM=(A15,75)
//ST2 EXEC PGM=PROG2,COND=(4,LE)
```

б) задание, содержащее операторы

```
//S1 JOB REGION=50K
//ST EXEC PROC=P2,PARM.ST1=A15,
//          COND.ST2,REGION.ST2=20K
```

После вызова каталогизированной процедуры P2 задание примет вид:

```
//S1   JOB   REGION=50K
//ST1 EXEC PGM=PROG1,PARM=A15
//ST2 EXEC PGM=PROG2,REGION=20K
```

Чтобы изменить параметры оператора DD, необходимо к оператору вызова процедуры добавить оператор DD следующего вида:

//*i.j* DD $p_1 = a_1, p_2=a_2, \ldots, p_n=a_n$

где i — имя шага каталогизированной процедуры (оператора EXEC); j — имя оператора DD, входящего в состав шага i каталогизированной процедуры; p_k ($k=1, 2, \ldots, n$) — имя изменяемого параметра оператора DD с именем j; a_k — либо пусто, либо новое значение параметра p_k.

Если изменения вносятся в оператор DD, входящий в состав первого шага каталогизированной процедуры, то имя шага (i) можно не указывать. Если оператор DD с именем j, входящий в состав i-го шага каталогизированной процедуры, содержит параметр с именем p_k, то он либо удаляется из оператора DD (когда a_k пусто), либо его значение заменяется на a_k (когда a_k не пусто). Если не содержит, то в состав оператора DD добавляется параметр $p_k = a_k$.

Пример 3. Пусть заданы:

а) каталогизированная процедура P3, содержащая операторы

```
//ST1 EXEC PGM=PG1,PARM=5F12.7,REGION=20K
//N1 DD DSN=NAB1,DISP=OLD,UNIT=5050,
//        VOL=SER=D17
//ST2 EXEC PGM=PG2
//N2 DD SYSOUT=A
```

б) задание, содержащее операторы

```
//S1 JOB
//ST EXEC P3,REGION.ST1=,REGION.ST2=20K
//ST1.N1 DD DISP=(NEW,KEEP),SPACE=(TRK,35)
//ST2.N2 DD SYSOUT=,DSN=NAB1,DISP=OLD,
//             UNIT=5050,VOL=SER=D17
//
```

После вызова каталогизированной процедуры P3 задание примет вид:

```
//S1 JOB
//ST1 EXEC PGM=PG1,PARM=5F12.7
//N1   DD   DSN=NAB1,DISP=(NEW,KEEP),UNIT=5050,
//             VOL-SER=D17,SPACE=(TRK,35)
//ST2 EXEC PGM=PG2,REGION=20K
//N2 DD DSN=NAB1,DISP=OLD,
//        UNIT=5050,VOL=SER=D17
//
```

Такой же результат будет получен, если использовать задание следующего вида:

```
//S1 JOB
```

```
//ST EXEC P3,REGION.ST1=,REGION.ST2=20K
//N1 DD DISP=(NEW,KEEP),SPACE=(TRK,35)
//ST2.N2 DD SYSOUT=,DSN=NAB1,DISP=OLD,
//          UNIT=5050,VOL=SER=D17
//
```

Новое значение любого изменяемого параметра, кроме DCB нужно записывать полностью (со всеми необходимыми подпараметрами). В новом значении параметра DCB можно указывать только изменяемые, удаляемые и добавляемые подпараметры. Чтобы удалить параметр DCB, нужно удалить каждый из его подпараметров (параметр DCB нельзя записывать с пустыми значениями).

Пример 4. Пусть задана каталогизированная процедура PR1, содержащая операторы

```
//RED EXEC PGM=TRAN
//WR1 DD DSN=CAT,DISP=OLD,
//       DCB=(RECFM=FB,BLKSIZE=160,LRECL=80)
```

При вызове этой процедуры параметр DCB, входящий в состав оператора DD, нужно заменить на параметр

```
DCB=(RECFM=F,BLKSIZE=160)
```

Вызов процедуры с указанными выше изменениями можно осуществить операторами

```
//R1 EXEC PR1
//RED.WR1 DD DCB=(RECFM=F,LRECL=)
```

Если при вызове этой процедуры параметр DCB нужно удалить из оператора DD, то такой вызов можно осуществить операторами

```
//R1 EXEC PR1
//RED.WR1 DD DCB=(RECFM=,BLKSIZE=,LRECL=)
```

Чтобы включить оператор DD в состав некоторого шага каталогизированной процедуры, необходимо к оператору вызова этой процедуры добавить оператор DD следующего вида:

```
//t.j DD p
```

где *t* — имя шага каталогизированной процедуры, в состав которого включается новый оператор DD; *j* — имя включаемого оператора DD; *p* — совокупность параметров включаемого оператора DD. Имя *j* включаемого оператора DD должно отличаться от всех имен операторов DD, входящих в шаг *t* каталогизированной процедуры.

Имя шага *t* можно не указывать, если оно относится к первому шагу процедуры.

Если в каталогизированной процедуре нужно модернизировать несколько шагов, то указания об их модификации должны записываться в порядке возрастания номеров этих шагов (имеются в виду порядковые номера шагов внутри задания). Указания об изменении параметров операторов DD внутри одного шага записываются в порядке возрастания номеров этих операторов внутри шага (здесь также имеются в виду порядковые номера операторов). После этого записываются указания о добавлении операторов DD к этому шагу.

Пример 5. Пусть задана каталогизированная процедура PR2, содержащая управляющие операторы

```
//Z1 EXEC PGM=FORM1
//FT01F001 DD UNIT=181
//FT02F001 DD UNIT=182
//Z2 EXEC PGM=FORM2
//FT03F001 DD DSN=N,DISP=OLD,UNIT=5050,
//            VOL=SER=D145
```

При вызове этой процедуры ее нужно модифицировать следующим образом: в операторы DD, входящие в состав первого шага (шага с именем Z1), нужно добавить параметр DCB=BLKSIZE=80; из оператора DD, входящего в состав второго шага задания (шага с именем Z2), нужно удалить параметр UNIT; в состав первого шага нужно включить оператор

```
//FT06F001 DD SYSOUT=A
```

Вызов этой процедуры с перечисленными выше изменениями можно осуществить операторами

```
//S1          EXEC PR2
//Z1.FT01F001 DD   DCB=BLKSIZE=80
//Z1.FT02F001 DD   DCB=BLKSIZE=80
//Z1.FT06F001 DD   SYSOUT=A
//Z2.FT03F001 DD   UNIT=
```

Вопросы и упражнения

1. Для чего предназначены каталогизированные процедуры?
2. Дать определение каталогизированной процедуры.
3. Написать оператор вызова каталогизированной процедуры KP1.

4. Пусть каталогизированная процедура KP2 содержит операторы:

```
//P1 EXEC PGM=LIB1,PARM=1
//D1   DD DSN=D1,DISP=(NEW,PASS),
//          UNIT=SYSSQ,SPACE=(CYL,5)
//P2 EXEC PGM=LIB2
//D2 DD SYSOUT=A
```

Написать управляющие операторы, которые осуществляют:

а) вызов каталогизированной процедуры KP2 без ее модификации;

б) вызов каталогизированной процедуры KP2 и замену в операторе EXEC, принадлежащем шагу P1, параметра PARM=1 на параметр PARM=2;

в) вызов каталогизированной процедуры KP2, удаление из оператора EXEC, принадлежащего шагу P1, параметра PARM и включение в оператор EXEC, принадлежащий шагу P2, параметра REGION=75K.

г) вызов каталогизированной процедуры KP2 и замену в операторе DD, принадлежащем шагу P1, параметра UNIT=SYSSQ на параметр UNIT=5050.

д) вызов каталогизированной процедуры KP2, замену в операторе DD, принадлежащем шагу P1, параметра DISP=(NEW,PASS) на параметр DISP=NEW, включение в оператор EXEC, принадлежащий шагу P2, нового параметра PARM=A и включение в шаг P2 нового оператора

```
//D3 DD SYSOUT=B.
```

10.2. Трансляция фортран-программы

Операционная система ОС ЕС ЭВМ включает в свой состав два транслятора с языка фортран: транслятор уровня G и транслятор уровня H. Транслятор уровня G относится к категории отладочных трансляторов. Он позволяет включать в транслируемые программы специальные отладочные средства, имеет высокую скорость трансляции, однако эффективность получаемых программ не высока. Транслятор уровня H относится к категории оптимизирующих трансляторов. Он позволяет получать высокоэффективные программы, однако имеет малую скорость трансляции и не использует никаких отладочных средств. Поэтому транслятор уровня H обычно применяют для трансляции уже отлаженных программ, которые часто выполняются и требуют для своего выполнения много вре-

мени. На практике наибольшее применение находит транслятор уровня G, с работой которого познакомимся более детально.

Фортран-программы пишутся на специальных бланках, разделенных на строки, каждая из которых содержит по 72 колонки. Каждый оператор программы начинается с новой строки и может занимать одну или несколько строк. Позиции с 1-й по 5-ю отводятся под метки операторов, шестая позиция отводится под символ продолжения оператора, позиции с 7-й по 72-ю отводятся под символы собственно оператора. Если оператор занимает несколько строк бланка, то метка оператора помещается в первую из этих строк. Поля меток во всех последующих строках оператора должны содержать пробелы. Шестая позиция первой строки оператора должна содержать пробел, шестая позиция остальных строк оператора — любой символ, отличный от пробела, который воспринимается транслятором как признак продолжения оператора.

Между операторами программы могут вставляться строки комментариев. Строки комментария должна начинаться с буквы С, которая записывается в первую колонку бланка. Остальные колонки строки комментария могут содержать любые символы. Строки комментариев транслятором не обрабатываются и в состав объектных модулей не включаются. Для обеспечения ввода программы в ЭВМ она обычно переносится с бланков на перфокарты с помощью перфорационных устройств, причем каждая строка бланка переносится на отдельную перфокарту.

Транслятор уровня G накладывает определенные ограничения на фортран-программы. Эти ограничения сводятся к следующему:

1) Число вложенных друг в друга операторов циклов и циклических элементов в списках вводимых и выводимых величин не должно превышать 25.

2) Число вложенных друг в друга арифметических и логических выражений (считая по открытым и закрытым скобкам) не должно превышать 100.

3) Число вложенных друг в друга указателей функций не должно превышать 25.

4) Максимальный объем транслируемой программы, вообще говоря, зависит от объема оперативной памяти, выделяемой транслятору на время его работы. Обычно транслятору выделяется 100К байтов оперативной памяти. В этом случае объем программы не должен превышать 400 строк (перфокарт).

5) Число строк (перфокарт), занимаемых одним оператором, не должно превышать 20.

6) Число строк комментариев, заключенных между двумя операторами, не должно превышать 30. Число строк комментариев, расположенных перед первым оператором программы, не ограничивается.

7) В операторе формата числа, означающие повторитель формата и ширину поля (w), не должны превышать 255.

Транслятор относится к системным обслуживающим программам и хранится в системной библиотеке загрузочных модулей. Имя транслятора — IEYFORT. Основными функциями транслятора являются:

а) ввод исходной программы (фортран-программы) в оперативную память;

б) синтаксический контроль исходной программы;

в) распечатка на АЦПУ исходной программы с указанием обнаруженных ошибок;

г) трансляция программы (перевод программы на язык объектных модулей);

д) распечатка на АЦПУ таблицы распределения памяти под переменные и массивы;

е) вывод объектной программы на перфокарты;

ж) перевод объектной программы на язык ассемблера и распечатка ее на АЦПУ;

з) вывод объектной программы в набор данных на магнитном диске или магнитной ленте.

Указания на выполнение транслятором тех или иных функций передаются транслятору через параметр PARM оператора EXEC.

Фортран-программа, подлежащая трансляции за один прием, может содержать один или несколько исходных модулей таких, как основная программа, подпрограмма-функция, подпрограмма-процедура.

Используемые в трансляторе наборы данных и их характеристики приведены в табл. 10.1. В этой таблице под DD-именем набора данных понимается имя оператора DD, описывающего этот набор. При описании этих наборов форматы блоков и длины записей указывать не требуется. Длины блоков необходимо указывать всегда, кроме случаев, когда ввод и вывод осуществляется через входные и выходные потоки, и когда ввод осуществляется из старых наборов, метки которых содержат необходимые длины блоков.

Ввод фортран-программы для трансляции чаще всего осуществляется с перфокарт через входной поток. В некоторых случаях фортран-программы с перфокарт переписываются на магнитные ленты и магнитные диски в последовательные наборы данных.

Таблица 10.1. Характеристики наборов данных, используемых транслятором уровня G

Назначение набора данных	DD-имя набора	Тип устройств ввода-вывода	Формат блоков	Длина блока	Длина записи
Ввод исходной программы	SYSIN	Устройство ввода с перфокарт	FB	80	80
		Магнитные диски, магнитные ленты	FB	$n \cdot 80$	80
Распечатка исходной программы, таблицы распределения памяти, объектной программы	SYSPRINT	АЦПУ	FBA	120	120
Вывод объектной программы на перфокарты	SYSPUNCH	Устройство вывода на перфокарты	FB	80	80
Вывод объектной программы на магнитный диск или магнитную ленту	SYSLIN	Дисковод, магнитофон	FB	80	80

В этих случаях ввод программ для трансляции осуществляется из этих наборов.

Рассмотрим несколько примеров записи оператора DD, описывающего входной набор транслятора.

Пример 1. Пусть исходная программа вводится для трансляции с перфокарт. Тогда оператор DD, описывающий входной набор данных, можно записать в следующем виде:

```
//SYSIN DD *
```

Пример 2. Пусть исходная программа вводится для трансляции с перфокарт через устройство, которое не определено как системное и имеет физический адрес 00A. Тогда оператор DD, описывающий входной набор данных, будет выглядеть так:

```
//SYSIN DD UNIT=00A,DCB=BLKSIZE=80
```

Пример 3. Пусть исходная программа вводится для трансляции с магнитного диска типа 5050, имеющего серийный номер D121. Набор, содержащий исходную программу, имеет следующие

характеристики: имя набора — MODFOR, формат блоков набора — FB, длина записей — 80 байтов, длина блоков — 240 байтов. Тогда оператор DD, описывающий этот набор, будет выглядеть так:

```
//SYSIN DD DSN=MODFOR,DISP=OLD,
//         UNIT=5050,VOL=SER=D121
```

В процессе трансляции каждый модуль исходной программы переводится в отдельный объектный модуль. При этом корневому объектному модулю (полученному из основной программы) присваивается имя MAIN, а остальным объектным модулям присваиваются имена подпрограмм, из которых эти модули были получены. Имена объектных модулей указываются в распечатках этих модулей, выдаваемых транслятором на АЦПУ. Полученная объектная программа в виде совокупности объектных модулей может выдаваться на перфокарты и выводиться в последовательный набор данных, содержащийся на магнитной ленте или на магнитном диске.

Выдача объектной программы на перфокарты чаще всего осуществляется через выходной поток. В этом случае оператор DD, описывающий набор с DD-именем SYSPUNCH, должен выглядеть так:

```
//SYSPUNCH DD SYSOUT=B
```

Возможна выдача объектной программы на перфокарты через выходной перфоратор, который не определен как системное устройство. В этом случае оператор DD для набора с именем SYSPUNCH должен выглядеть так:

```
//SYSPUNCH DD UNIT=α,DCB=BLKSIZE=80
```

где α — шифр выходного перфоратора.

В практической работе выдачей объектной программы на перфокарты пользуются редко.

Основным набором, в который помещается объектная программа, является набор с DD-именем SYSLIN, размещаемый на магнитном диске или магнитной ленте. Этот набор используется редактором связей, из которого он берет объектные модули для редактирования и объединения в загрузочный модуль. Рассмотрим несколько примеров описания такого набора.

Пример 1. Объектная программа помещается в новый набор данных, который создается на диске типа 5050 с серийным номером 000222. После выполнения трансляции и всего задания набор необходимо сохранить. Оператор DD, описывающий этот набор, может выглядеть так:

```
//SYSLIN DD DSN=N1,DISP=(NEW,KEEP),
```

```
// UNIT=5050,VOL=SER=000222,
// SPACE=(80,200),DCB=BLKSIZE=80
```

Пример 2. Объектная программа помещается в новый набор данных, который создается на диске типа 5050. После выполнения трансляции набор необходимо сохранить для использования в последующих шагах, а после выполнения всего задания — уничтожить. Оператор DD, описывающий этот набор, может выглядеть так:

```
//SYSLIN DD DSN=&N2,DISP=(NEW,PASS),
//  UNIT=5050,SPACE=(TRK,100),
//  DCB=BLKSIZE=80
```

Пример 3. Объектные модули, получаемые в результате трансляции, добавляются к ранее полученным объектным модулям, содержащимся в некотором наборе данных на диске типа 5061 с серийным номером D115. После выполнения транслятора и всего задания набор требуется сохранить. Оператор DD, описывающий этот набор, может выглядеть так:

```
//SYSLIN DD DSN=N3,DISP=MOD, UNIT=5051,
// VOLUME=SER=D115
```

Распечатка сообщений транслятора (исходной программы, таблицы распределения памяти и объектной программы на языке ассемблера) на АЦПУ чаще всего осуществляется через выходную очередь. В этом случае выходной набор с DD-именем SYSPRINT будет выглядеть так:

```
//SYSPRINT DD SYSOUT=A
```

Если же распечатка осуществляется через АЦПУ, которое не определено как системное выходное устройство, то оператор DD для этого набора будет выглядеть так:

```
//SYSPRINT DD UNIT=α,DCB=BLKSIZE=129
```

где α — код АЦПУ.

Распечатка исходной программы осуществляется в виде таблицы, содержащей три колонки. В первой колонке печатаются четырехзначные порядковые номера операторов, во второй — метки операторов, в третьей — собственно операторы. Пример такой распечатки приведен на рис. 10.1.

Если в каком-либо операторе программы транслятор обнаружил одну или несколько синтаксических ошибок, то в строке, следующей

за этим оператором, размещаются указатели ошибок (символы ¤), а в последующих строках — сообщения об ошибках. Указатель ошибки размещается в той позиции таблицы, в которой была обнаружена ошибка. Сообщение об ошибке имеет вид:

n) IHC*m*I *t*

где *n* — порядковый номер ошибки в операторе; *m* — трехзначное целое число, означающее *код ошибки*; *t* — пояснительный текст.

```
0001       X=2.0
0002       Y=2.5
0003       CALL RV2(X, Y, &2, &3)
0004     2 X=X**2
0005       GO TO 6
0006     3 Y=Y**2
0007     6 CALL RV2(X, Y, &4, &5)
0008     4 X=2*X
0009       GO TO 7
0010     5 Y=2*Y
0011     7 STOP
0012       END
```

Рис. 10.1. Пример распечатки исходной программы.

Рассмотрим такой пример: пусть в программе содержится оператор FORMAT(I3,F5.3,5HTAB). В этом операторе имеются две синтаксические ошибки (отсутствует метка оператора и указатель длины текстовой константы 5HTAB не соответствует ее длине). Этот оператор будет распечатан в следующем виде:

```
      FORMAT (I3,F5.3,5HTAB)
¤                          ¤
01) IHC002I LABEL 02) IHC013I SYNTAX
```

Расшифровка кодов ошибок содержится в табл. 10.2.

Сообщения об ошибках, относящихся ко всей программе, размещаются после операторов программы.

Распечатками сведений о распределении памяти, а также распечатками объектной программы разработчики фортран-программы в своей практической работе обычно не пользуются. Эти распечатки анализируются системными программистами при возникновении подозрений в правильности работы транслятора или других

Таблица 10.2. Коды ошибок, выдаваемые транслятором уровня G

Код ошибки	Код возврата	Причина
001	8	Тип константы, переменной или выражения не соответствует их использованию.
002	0	Оператор, на который возможна ссылка, не помечен.
003	0	Символическое имя содержит более 6 символов.
004	0	В операторе пропущена запятая.
005	8	Метка оператора используется неправильно.
006	8	Метка оператора не уникальна.
007,008	8	Символическое имя используется не в соответствии с его определением.
009	8	Неверный порядок следования операторов.
010	8	Число выходит за пределы допустимой области значений.
011	8	Используется элемент массива, имя которого не было определено.
012	8	Количество индексов, используемых для указания элемента массива, не соответствует размерности массива или размерность массива больше 7.
013	8	Формат оператора не соответствует синтаксису языка фортран.
015	0	Отсутствует оператор END в исходной программе.
016	8	Неправильное использование оператора.
017	0	Предупреждение о неправильном использовании оператора.
018	4	Количество аргументов, указанных при обращении к программе библиотеки фортрана, неверное.
019	4	Имени функции или имени входа не присвоено значение в подпрограмме FUNCTION.
020	4	Переменным или массивам из именованной общей области не может быть распределена память из-за ошибок.
021	8	В программе есть незакрытые циклы DO.
022	8	Указанные метки операторов не определены в исходной программе.

Таблица 10.2. (продолжение)

Код ошибки	Код возврата	Причина
023	4	Переменным не может быть распределена память из-за противоречия между группами эквивалентности.
024	4	В операторе эквивалентности используется элемент массива, который не определен в программе.
025	4	Ошибки при определении массива с переменными измерениями.
026	4	В программе BLOCK DATA присваиваются начальные значения переменным, не принадлежащим именованной общей области.
027	8	Оператор содержит более 19 строк продолжения.
028	16	Трансляция прекращается из-за недостатка памяти для размещения таблиц.
029	—	Произошла ошибка во время перфорации объектного модуля при задании режима DECK.
030	16	Произошла ошибка во время построения модуля при задании режима LOAD.
031	16	Превышен размер таблицы, используемой транслятором.
032	0	Оператор конца набора данных /* предшествует операторам фортрана.
033	0	Количество строк с комментариями между двумя операторами превышает 30.
034	8/16	Произошла ошибка ввода-вывода во время трансляции.
035	—	Невозможно использовать набор данных из-за отсутствия оператора DD.
036	0	Невыполнимый оператор имеет метку.
037	4	Размерность массива определяется более чем один раз.
039	0	В подпрограмме отсутствует оператор RETURN.
040	8	Ошибка в подпрограмме BLOCK DATA.

элементов операционной системы. Поэтому форма этих распечаток нами рассматриваться не будет.

Вызов транслятора с указанием режимов его работы осуществляется оператором EXEC, который должен иметь вид:

$$//t\ \text{EXEC PGM=IEYFORT,PARM=}(\alpha_1,\alpha_2,\ \ldots,\ \alpha_n),\beta$$

где t — имя оператора EXEC; $\alpha_i (i = 1,2,\ \ldots,n)$ — подпараметры параметра PARM, определяющие режимы работы транслятора; β — совокупность прочих параметров оператора EXEC.

Основные режимы работы транслятора приведены в табл. 10.3.

Внутри параметра PARM его подпараметры могут следовать в произвольном порядке. Если какой-либо из подпараметров не указан, то будет браться его стандартное значение. Стандартные значения подпараметров устанавливаются в процессе генерации операционной системы. Если отсутствует параметр PARM, то значения всех его подпараметров берутся стандартными.

Рассмотрим пример.

Пусть задание на трансляцию фортран-программы выглядит так:

```
//TRAN JOB
//S1 EXEC PGM=IEYFORT,PARM=(NODECK,
//  LOAD,SOURCE,NOMAP,NOLIST),REGION=100K
//SYSIN DD *
  (Исходная фортран-программа)
/*
//SYSPRINT DD SYSOUT=A
//SYSLIN DD DSN=B1,DISP=(NEW,KEEP),
//             UNIT=5050,VOL=SER=D44,
//             SPACE=(80,(200,100)),DCB=BLKSIZE=80
```

Объектные модули, созданные в результате выполнения этого задания, будут выданы в основной выходной набор данных (с DD-именем SYSLIN), который будет создан на магнитном диске с серийным номером D44. На АЦПУ (в выходной набор с DD-именем SYSPRINT) будет выдана только исходная программа с возможными сообщениями об ошибках.

Вопросы и упражнения

1. Назвать основные функции транслятора.
2. Написать DD-имена наборов данных, используемых транслятором. Для чего эти наборы предназначены?

Таблица 10.3. Основные режимы работы транслятора уровня G

Условный номер подпараметра	Значение подпараметра	Режим
1	DECK	Вывод в набор данных с DD-именем SYSPUNCH разрешен.
	NODECK	Вывод в набор данных с DD-именем SYSPUNCH не разрешен.
2	LOAD	Вывод в набор данных с DD-именем SYSLIN разрешен.
	NOLOAD	Вывод в набор данных с DD-именем SYSLIN не разрешен.
3	SOURCE	В набор данных с DD-именем SYSPRINT разрешен вывод исходной программы с сообщениями об ошибках.
	NOSOURCE	В набор данных с DD-именем SYSPRINT разрешен вывод сообщений об ошибках в исходной программе. Вывод исходной программы не разрешен.
4	MAP	Вывод в набор данных с DD-именем SYSPINT таблицы распределения памяти разрешен.
	NOMAP	Вывод в набор данных с DD-именем SYSPRINT таблицы распределения памяти не разрешен.
5	LIST	Вывод в набор данных с DD-именем SYSPRINT распечатки объектных модулей разрешен.
	NOLIST	Вывод в набор данных с DD-именем SYSPRINT распечатки объектных модулей не разрешен.

3. Какие имена присваиваются объектным модулям?

4. Составить задание на трансляцию фортран-программы, в которой предусмотреть: а) выдачу объектных модулей программы на перфокарты; б) распечатку на АЦПУ сообщений об ошибках в программе.

10.3. Редактирование программ

Основной задачей этапа редактирования программы является объединение отдельных объектных модулей, полученных на этапе трансляции, в единую программу, называемую загрузочным модулем.

Редактирование осуществляется специальной программой, называемой редактором связей. Необходимо сразу же оговориться, что в состав вновь создаваемого загрузочного модуля могут включаться не только объектные модули, но и ранее созданные загрузочные модули. Редактор связей может создавать загрузочные модули простой структуры и загрузочные модули оверлейной структуры (с запланированным перекрытием).

Загрузочный модуль простой структуры является монолитом, который при его исполнении загружается в оперативную память целиком. *Загрузочный модуль оверлейной структуры* создается в виде совокупности сегментов, которые при исполнении загрузочного модуля вызываются в оперативную память в определенной последовательности и могут перекрывать друг друга (использовать одни и те же участки оперативной памяти). Такая структура позволяет большие по объему программы выполнять на небольших участках оперативной памяти.

Редактор связей относится к системным обслуживающим программам, хранится в системной библиотеке загрузочных модулей SYS1.LINKLIB и имеет имя IEWL.

Основными функциями редактора связей являются:

а) ввод в оперативную память исходных объектных и загрузочных модулей;

б) формирование из исходных модулей единой программы в виде загрузочного модуля заданной структуры;

в) составление и распечатка таблицы распределения памяти (плана загрузочного модуля);

г) составление и распечатка таблицы внешних ссылок (адресов обращений одних программных модулей к другим);

д) распечатка управляющих операторов редактора связей;

е) вывод полученного загрузочного модуля.

Ввод исходных объектных модулей может осуществляться из основного входного набора данных последовательного доступа с DD-именем SYSLIN, а также из дополнительных входных наборов данных последовательного и библиотечного доступов. Ввод исходных загрузочных модулей может осуществляться только из дополнительных входных наборов библиотечного доступа.

Если в загрузочном модуле (после включения в него всех исходных модулей из основного и дополнительных входных наборов) окажутся неразрешенные ссылки (обращения к модулям, которых не оказалось в основном и дополнительных наборах), то редактор связей может разрешать эти ссылки автоматически путем включения в формируемый модуль объектных и загрузочных модулей из основной библиотеки с DD-именем SYSLIB и дополнительных библиотек. В качестве основной библиотеки обычно используется библиотека фортрана SYS1.FORTLIB, которая содержит загрузочные модули, предназначенные для реализации в фортран-программах математических функций и операций ввода-вывода.

Распечатка таблицы распределения памяти, таблицы внешних ссылок и управляющих операторов редактора связей осуществляется на АЦПУ через выходной набор с DD-именем SYSPRINT. Обычно этими распечатками пользуются системные программисты, обслуживающие операционную систему и хорошо знающие все детали работы транслятора и редактора связей. Набор SYSPRINT используется редактором связей и для выдачи сообщений об ошибках.

Вывод полученного загрузочного модуля осуществляется в библиотечный набор данных с DD-именем SYSLMOD.

Характеристики основных наборов данных, используемых редактором связей, представлены в табл. 10.4. В эту таблицу включен набор с DD-именем SYSUT1, который используется как рабочий набор (для размещения промежуточных результатов работы редактора связей).

Дополнительные входные наборы последовательного доступа должны иметь такие же характеристики, как и набор с DD-именем SYSLIN, а дополнительные входные наборы библиотечного доступа — как набор с DD-именем SYSLIB. Операторы DD для вновь создаваемых наборов данных на магнитных дисках и лентах должны содержать параметр DCB с подпараметром BLKSIZE. Характеристики наборов данных, указанные в табл. 10.4, удовлетворяют требованиям всех существующих версий редактора связей. Редактор связей обладает большими возможностями по формированию загрузочных модулей различной структуры и имеет сложное управление. Нами будут рассмотрены лишь основные его возможности, связанные с формированием одного загрузочного модуля простой структуры.

Управление работой редактора связи осуществляется через параметр PARM управляющего оператора EXEC и через специальные управляющие операторы редактора связей.

Таблица 10.4. Характеристики наборов данных, используемых редактором связей

DD-имя набора	Тип устройства ввода-вывода	Формат блоков	Длина блоков	Длина записи
SYSLIN	Устройство ввода с перфокарт, магнитные диски, магнитные ленты	F	80	80
SYSLIB (для объектных модулей)	Магнитные диски	F	80	80
SYSLIB (для загрузочных модулей)	Магнитные диски	U	1024	1024
SYSPRINT	АЦПУ	UA	121	121
SYSUT1	Магнитные диски	U	1024	1024
SYSLMOD	Магнитные диски	U	1024	1024

Параметр PARM используется для указания режимов работы редактора связей, которые включаются в него как подпараметры. Основные из этих режимов означают следующее:

MAP — распечатать таблицу распределения памяти;

XREF — распечатать таблицу распределения памяти и таблицу внешних ссылок;

LIST — распечатать управляющие операторы редактора связей;

NCAL — автоматическое разрешение внешних ссылок не осуществлять;

LET — приписать формируемому загрузочному модулю характеристику «выполнимый» даже, если не все внешние ссылки разрешены.

Каждый из этих режимов имеет либо утвердительную, либо отрицательную форму. Противоположная форма принимается по умолчанию. Так, например, отсутствие подпараметра MAP говорит о том, что составлять и распечатывать таблицу распределения памяти не нужно, отсутствие LET говорит о том, что при наличии в загрузочном модуле неразрешенных ссылок ему приписывается характеристика «не выполнимый».

NAME должен размещаться после всех объектных модулей и после всех управляющих операторов. В процессе работы редактора связей в формируемый загрузочный модуль вначале включаются исходные модули из основного входного набора, расположенные перед первым оператором INCLUDE, затем — исходные модули из наборов данных, перечисленных в первом операторе INCLUDE, затем — исходные модули из основного входного набора, расположенные перед вторым оператором INCLUDE и т. д. Если в сформированном таким образом загрузочном модуле окажутся неразрешенные внешние ссылки, то будет выполняться процедура автоматического разрешения ссылок (если она не запрещена путем указания соответствующего режима). При этом вначале просматриваются библиотеки, перечисленные в управляющих операторах LIBRARY, а затем библиотека с DD-именем SYSLIB. Сообщения об обнаруженных ошибках редактор связей выдает в набор данных с DD-именем SYSPRINT в следующем виде:

IEW*m t*

где *m* — номер ошибки, *t* — пояснительный текст. Расшифровка кодов ошибок, выдаваемых редактором связей, содержится в табл. 10.5.

Рассмотрим несколько примеров на составление заданий для редактора связей.

Пример 1. Пусть требуется отредактировать программу, состоящую из основной программы с именем MAIN и двух подпрограмм-процедур P1 и P2, которые предварительно оттранслированы и в виде объектных модулей выданы на перфокарты. Задание на редактирование такой программы можно составить в следующем виде:

```
//RED JOB
//  EXEC PGM=IEWL
//SYSLIB DD DSN=SYS1.FORTLIB,DISP=SHR
//SYSPRINT DD SYSOUT=A
//SYSUT1   DD UNIT=SYSDA,SPACE=(1024,(20,10)),
//              DCB=BLKSIZE=1024
//SYSLMOD DD DSN=&FFL,DISP=NEW,UNIT=SYSDA,
//         SPACE=(1024,(20,10,1)),DCB=BLKSIZE=1024
//SYSLIN DD *
     a1
     a2
     a3
     a4
/*
//
```

где α1 — перфокарты с основной программой, α2 — перфокарты с подпрограммой P1, α3 — перфокарты с подпрограммой P2, α4 — управляющий оператор редактора связей NAME=PROG1.

При выполнении этого задания для редактора связей будет отведено поле оперативной памяти стандартного объема. Исходные объектные модули будут вводиться через входной поток (с перфокарт) и включаться в загрузочный модуль в следующем порядке: модуль MAIN (разместится первым), модуль P1 (разместится вторым) и модуль P2 (разместится третьим). Если в загрузочном модуле окажутся неразрешенные ссылки, то редактор будет пытаться их разрешить путем включения в него недостающих модулей из библиотеки SYS1·FORTLIB, сведения о которой содержатся в системном каталоге наборов данных.

В качестве точки входа в загрузочный модуль будет принята точка входа в первый загрузочный модуль, т. е. точка входа в основную программу MAIN. Загрузочный модуль поместится в разделе PROG1 временной библиотеки &FFL, которая будет создана на одном из магнитных дисков, подключенном к ЭВМ.

Пример 2. Пусть требуется отредактировать программу, описанную в примере 1. Модули MAIN, P1 и P2 предварительно оттранслированы и в виде объектных модулей помещены в набор данных последовательного доступа, имеющий имя INLIN и расположенный на магнитном диске типа 5050 с серийным номером D125. Модули размещаются в наборе в следующем порядке: P1 (на первом месте), P2 (на втором месте), MAIN (на третьем месте).

Задание на редактирование этой программы может выглядеть таким же образом, как и задание, приведенное в примере 1. Однако оператор DD с именем SYSLIN должен быть другим, например, таким:

```
//SYSLIN DD  DSN=INLIN, DISP=OLD,
//              UNIT=5050,VOL=SER=D125
//         DD *
     α1
     α2
/*
```

где α1 — управляющий оператор редактора связей ENTRY MAIN; α2 — управляющий оператор редактора связей NAME PROG1. Если управляющий оператор редактора связей ENTRY MAIN опустить, то в качестве точки входа в загрузочный модуль PROG1 будет принята точка входа в подпрограмму P1.

Управляющие операторы используются для передачи редактору связей информации об исходных и формируемых модулях. Они пишутся на таких же бланках, что и операторы заданий. Каждый управляющий оператор должен начинаться с новой строки. Первая позиция этой строки должна содержать пробел, за которым располагаются имя операции, пробел и список параметров, разделенных запятыми. Список параметров не должен выходить за 71-ю колонку. При необходимости его можно перенести на следующую строку. Прерванный список параметров должен заканчиваться запятой, а в 72-ю колонку помещается знак переноса. В качестве знака переноса можно использовать любой символ, отличный от пробела. На следующей строке список параметров продолжается, начиная с колонки 16. В редакторе связей предусмотрено большое количество управляющих операторов, которые могут вводиться из основного и дополнительных входных наборов данных. Однако нами будет рассмотрена только основная часть этих операторов и основной путь их поступления — через входной набор данных с DD-именем SYSLIN. К таким операторам относятся:

ENTRY, INCLUDE, LIBRARY, NAME.

Управляющий оператор ENTRY используется для указания точки входа в загрузочный модуль, т. е. точки, в которую передается управление после вызова загрузочного модуля на исполнение. Этот оператор имеет вид

ENTRY *i*

где *i* — имя программного модуля, который является корневым по отношению к другим программным модулям, входящим в состав загрузочного модуля. Как правило, таким модулем является основная программа с именем MAIN.

Управляющий оператор ENTRY обязательно должен использоваться только в том случае, когда корневой программный модуль (основная программа) не является первым в загрузочном модуле.

Управляющий оператор INCLUDE служит для указания дополнительных наборов данных, которые используются редактором связей для ввода исходных объектных и загрузочных модулей.

Этот оператор записывается в следующем виде:

INCLUDE $c_1, c_2, \ldots, c_n$

где c_i ($i = 1, 2, \ldots, n$) — ссылки на набор данных.

В качестве ссылки на набор данных последовательного доступа используется DD-имя этого набора. Ссылка на набор данных библио-

течного доступа задается в следующем виде:

$$d\ (r_1, r_2, \ldots, r_k)$$

где d — DD-имя библиотечного набора данных; r_j $(j = 1, 2, \ldots, k)$ имена используемых разделов набора данных d.

В каждом наборе последовательного доступа может содержаться несколько объектных модулей, и все они будут включены в состав формируемого загрузочного модуля. В каждом разделе библиотечного набора данных может содержаться по одному объектному или загрузочному модулю. В состав формируемого загрузочного модуля будут включены те из них, которые содержатся в перечисленных разделах.

Пример оператора INCLUDE:

```
INCLUDE D1,D2,LIB1(PROG1,PROG2)
```

Управляющий оператор LIBRARY используется для указания дополнительных библиотек, которые используются в процессе автоматического разрешения оставшихся неразрешенных внешних ссылок. Этот оператор записывается в следующем виде:

$$\text{LIBRARY}\ \ c_1, c_2, \ldots, c_n$$

где c_i $(i = 1, 2, \ldots, n)$ — ссылки на библиотечные наборы данных. Ссылки имеют вид:

$$d\ (r_1, r_2, \ldots, r_k)$$

где d — DD-имя библиотечного набора; r_j $(j = 1, 2, \ldots, k)$ — имена используемых разделов набора d.

Управляющий оператор NAME используется для указания имени формируемого загрузочного модуля. Он записывается в следующем виде:

$$\text{NAME}\ \ i$$

где i — имя загрузочного модуля (имя раздела выходного библиотечного набора с DD-именем SYSLMOD, в который будет помещен загрузочный модуль).

Если вновь создаваемый загрузочный модуль должен заменить уже содержащийся в наборе SYSLMOD загрузочный модуль с именем i, то этот оператор необходимо записать в форме NAME i (R).

Если параметр DSNAME оператора DD, описывающего набор SYSLMOD, уже содержит имя раздела, то управляющий оператор NAME можно не использовать.

Управляющие операторы ENTRY, INCLUDE и LIBRARY могут размещаться в любом месте основного входного набора данных (до объектных модулей, между ними и после них). Оператор

Набор данных INLIN, содержащий объектные модули P1, P2 и MAIN, можно описать в задании как дополнительный входной набор редактора связей. Тогда задание на редактирование программы можно представить в следующем виде:

```
//RED JOB
//EXEC PGM=IEWL
//SYSLIB DD DSN=SYS1.FORTLIB,DISP=SHR
//SYSPRINT DD SYSOUT=A
//SYSUT1 DD UNIT=SYSDA,SPACE=(1024,(20,10)),
//          DCB=BLKSIZE=1024
//SYSLMOD DD DSN=&FFL,DISP=NEW,
//           UNIT=SYSDA,SPACE=(1024,(20,10,1)),
//           DCB=BLKSIZE=1024
//IN1 DD DSN=INLIN,DISP=OLD,
//       UNIT=5050,VOL=SER=D125
//SYSLIN DD *
    α1
    α2
    α3
/*
//
```

где α1 — управляющий оператор редактора связей INCLUDE IN1; α2 — управляющий оператор редактора связей ENTRY MAIN; α3 — управляющий оператор редактора связей NAME PROG1.

Имя загрузочного модуля (имя раздела библиотечного набора) можно указывать в параметре DSN оператора DD, описывающего этот набор. Если используется этот прием, то операторы DD с именами SYSLMOD и SYSLIN, входящие в приведенное выше задание, должны иметь вид:

```
//SYSLMOD DD DSN=&FFL(PROG1),DISP=NEW,
//           UNIT=SYSDA,SPACE=(1024,(20,10,1)),
//           DCB=BLKSIZE=1024
//SYSLIN DD *
    α1
    α2
/*
```

Вопросы и упражнения

1. Для чего предназначен редактор связей?

2. Написать DD-имена основных наборов данных, используемых редактором связей. Для чего эти наборы предназначены?

Таблица 10.5 Основные коды ошибок, выдаваемые редактором связей

Код ошибки	Причина
0053	Входная точка, указанная в операторе ENTRY, не является именем программного модуля.
0132	В загрузочном модуле остались неразрешенные внешние ссылки.
0212	За управляющим оператором редактора связей, содержащим символ переноса, нет продолжения этого оператора.
0264	Исходный модуль содержит слишком много внешних символов.
0284	Для набора данных, используемого редактором связей, отсутствует оператор DD.
0294	В наборе данных, DD-имя которого напечатано, указана слишком большая длина блока.
0302	В управляющем операторе редактора связей содержится ошибка.
0342	В операторе LIBRARY неверно указано имя раздела.
0382	Не хватает данных в исходном объектном модуле.
0394	Либо сбой в работе аппаратуры, либо под оглавление библиотечного набора данных не выделено места.
0421	Имя сформированного загрузочного модуля было ранее использовано.
0432	Отсутствует оператор DD, описывающий набор данных, указанный в операторе INCLUDE.
0492	Управляющий оператор NAME содержится не в основном входном наборе.
0502	Характеристики набора данных, указанные в операторе DD, отличаются от характеристик набора данных, указанных в операторе INCLUDE.
0522	Неверно задан формат блоков набора.
0532	Неверно задана длина блока набора данных.
0564	В операторе DD с именем SYSUT1 или SYSLMOD в параметре UNIT указано устройство, не приемлемое для редактора связей.
0574	В операторе DD с именем SYSLIN в параметре DCB указана нулевая длина блока.
0584	В операторе DD с именем SYSLIN неверно указан формат блоков.
0601	В параметре PARM указаны два противоречивых режима.
0622	В операторе DD для дополнительного входного набора в параметре DCB указана нулевая длина блока.

3. Каким образом редактору связей передается информация, используемая для управления его работой?

4. Назвать основные управляющие операторы редактора связей. Для чего они предназначены?

5. Пусть объектные модули MAIN (головной), S1, S2 и S3, включаемые в состав редактируемой программы, размещаются в следующих наборах данных: MAIN — в наборе N1; S1 и S2 — в наборе N2; S3 — в наборе N3. Программе необходимо присвоить имя PROG5 и поместить ее в заранее созданную библиотеку B1. Все наборы размещаются на магнитном диске типа 5050 с серийным номером S 12. Составить задание на редактирование этой программы. Для набора данных с DD-именем SYSUT1 достаточно отвести 20 блоков по 1024 байта.

10.4. Каталогизированные процедуры трансляции и редактирования

В библиотеке каталогизированных процедур операционной системы ОС ЕС ЭВМ содержится ряд процедур, которые упрощают работу по составлению заданий на трансляцию, редактирование и выполнение программ.

Процедура трансляции (FORTGC):

```
//FORT EXEC PGM=IEYFORT,REGION=100K
//SYSPRINT DD SYSOUT=A
//SYSPUNCH DD SYSOUT=B
//SYSLIN DD DSN=&LOADSET,DISP=(MOD,PASS),
//          UNIT=SYSSQ,SPACE=(80,(200,100),RLSE),
//          DCB=BLKSIZE=80
```

В этой процедуре оператор вызова транслятора параметра PARM не содержит, следовательно, режимы работы транслятора берутся стандартными. Как правило, стандартные режимы предусматривают вывод распечатки исходной программы и сообщений об ошибках в набор данных с DD-именем SYSPRINT, а также вывод объектной программы в основной выходной набор данных с DD-именем SYSLIN.

Процедура трансляции и редактирования (FORTGCL):

```
//FORT EXEC PGM=IEYFORT, REGION=100K
//SYSPRINT  DD SYSOUT=A
//SYSPUNCH DD SYSOUT=B
//SYSLIN DD DSN=&LOADSET,DISP=(MOD,PASS),
//          UNIT=SYSSQ,SPACE=(80,(200,100),RLSE),
```

```
//             DCB=BLKSIZE=80
//LKED EXEC PGM=IEWL,REGION=96K,PARM=
//             (XREF,LET,LIST),COND=(4,LT,FORT)
//SYSLIB DD DSN=SYS1.FORTLIB,DISP=SHR
//SYSPRINT DD SYSOUT=A
//SYSLIN DD DSN=&LOADSET,DISP=(OLD,DELETE)
//  DD DDNAME=SYSIN
//SYSUT1 DD UNIT=SYSDA, SPACE=(1024,(20, 10),RLSE),
//             DCB=BLKSIZE=1024,DSN=&SYSUT1
//SYSLMOD DD  DSN=&GOSET(MAIN),DISP=(NEW,PASS),
//               UNIT=SYSDA,SPACE=(1024,(20,10,1),RLSE)
```

Эта процедура содержит 2 шага, в первом из которых осуществляется вызов транслятора, а во втором — редактора связей. При использовании этой процедуры в первый ее шаг нужно добавлять оператор DD с именем SYSIN, описывающий входной набор данных транслятора, а во второй — оператор DD с именем SYSIN, описывающий основной входной набор данных редактора связей, через который будут вводиться управляющие операторы редактора связей. Если управляющие операторы редактора связей не потребуются, то такой оператор добавлять не нужно. Полученный загрузочный модуль помещается во временную библиотеку &GOSET под именем MAIN.

Процедура редактирования и выполнения (FORTGLG):

```
//LKED EXEC PGM=IEWL, REGION=96K,
//             PARM=(XREF,LET,LIST)
//SYSLIB DD DSN=SYS1.FORTLIB,DISP=SHR
//SYSLIN DD DDNAME=SYSIN
//SYSUT1  DD  UNIT=SYSDA,SPACE=(1024,(20,10),RLSE),
//             DCB=BLKSIZE=1024,DSN=&SYSUT1
//SYSLMOD DD DSN=&GOSET(MAIN),DISP=(NEW, PASS),
//  UNIT=SYSDA,SPACE=(1024,(20,10,1),RLSE),
//  DCB=BLKSIZE=1024
//SYSPRINT DD SYSOUT=A
//GO EXEC PGM=*.LKED.SYSLMOD,COND=(4,LT,LKED)
//FT05F001 DD DDNAME=SYSIN
//FT06F001 DD SYSOUT=A
//FT07F001 DD  SYSOUT=B
```

Эта процедура содержит два шага, в первом из которых осуществляется вызов редактора связей, а во втором — вызов того загрузочного модуля, который был получен на первом шаге. При ис-

пользовании этой процедуры в ее первый шаг нужно добавлять оператор DD с именем SYSIN, описывающий основной входной набор данных редактора связей, а во второй — оператор DD с именем SYSIN, описывающий входной набор данных загрузочного модуля (со ссылочным номером 5). Кроме того, во второй шаг нужно добавлять операторы DD, описывающие другие наборы данных загрузочного модуля и не содержащиеся в этом шаге каталогизированной процедуры.

Процедура трансляции, редактирования и выполнения (FORTGCLG):

```
//FORT EXEC PGM=IEYFORT,REGION=100K
//SYSPRINT DD SYSOUT=A
//SYSPUNCH DD SYSOUT=B
//SYSLIN DD DSN=&LOADSET,DISP=(MOD,PASS),
//             UNIT=SYSSQ,SPACE=(80,(200,100),RLSE),
//             DCB=BLKSIZE=80
//LKED EXEC PGM=IEWL,REGION=96K,PARM=(XREF,
//              LET,LIST),COND=(4,LT,FORT)
//SYSLIB DD DSN=SIS1.FORTLIB,DISP=SHR
//SYSLIN DD DSN=&LOADSET,DISP=(OLD,DELETE)
// DD DDNAME=SYSIN
//SYSUT1 DD UNIT=SYSDA,SPACE=(1024,(20,10),
//             RLSE),DCB=BLKSIZE=1024,DSN=&SYSUT1
//SYSPRINT DD SYSOUT=A
//SYSLMOD DD DSN=&GOSET(MAIN),DISP=(NEW,
//               PASS),UNIT=SYSDA,SPACE=(1024,
//               20,10,1),RLSE),DCB=BLKSIZE=1024
//GO EXEC PGM=*.LKED.SYSLMOD,
//          COND=((4,LT,FORT),(4,LT,LKED))
//FT05F001 DD DDNAME=SYSIN
//FT06F001 DD SYSOUT=A
//FT07F001 DD SYSOUT=B
```

Эта процедура содержит 3 шага, в первом из которых осуществляется вызов транслятора, во втором — редактора связей; в третьем — вызов загрузочного модуля, который был получен на втором шаге. При использовании этой процедуры в ее первый шаг нужно добавлять оператор DD с именем SYSIN, описывающий входной набор данных транслятора; во второй — оператор DD с именем SYSIN, описывающий основной входной набор данных редактора связей, через который будут вводиться управляющие операторы редактора связей; в третий — оператор DD с именем SYSIN,

описывающий входной набор данных загрузочного модуля (со ссылочным номером 5). Кроме того, в третий шаг нужно добавлять операторы DD, описывающие другие наборы данных загрузочного модуля и не содержащиеся в этом шаге каталогизированной процедуры.

Рассмотрим несколько примеров на использование каталогизированных процедур.

Пример 1. Пусть фортран-программа состоит из одной основной программы, которая нанесена на перфокарты. Требуется оттранслировать эту программу, поместить в набор данных с именем OBMOD, который нужно создать на магнитном диске 5050 с серийным номером S77 и сохранить для дальнейшего использования. Задание на эту работу с использованием каталогизированной процедуры FORTGC может выглядеть так:

```
//RAB  JOB
//  EXEC  PROC=FORTGC
//SYSLIN DD  DSN=OBMOD,DISP=(NEW,KEEP),
//    UNIT=5050,VOL=SER=S77,  SPACE=(80,100)
//SYSIN DD  *
(Исходная фортран-программа)
/*
//
```

Пример 2. Пусть фортран-программа состоит из одной основной программы, которая нанесена на перфокарты. Эта программа использует следующие наборы данных: входной перфокарточный набор со ссылочным номером 5, временный набор последовательного доступа, размещенный на магнитном диске, со ссылочным номером 8; выходной набор на бумажной ленте со ссылочным номером 6. Все эти наборы имеют стандартные характеристики. Требуется оттранслировать эту программу, отредактировать и выполнить. Задание на эти работы с использованием каталогизированной процедуры FORTGCLG может выглядеть так:

```
//GAMMA JOB
//   EXEC FORTGCLG
//FORT.SYSIN DD *
(Исходная программа)
/*
//GO.SYSIN DD *
     (Входные данные рабочей программы)
/*
//GO.FT08F001 DD DSN=&N1,DISP=NEW,
```

```
//                    UNIT=SYSDA,SPACE=(CYL,50)
//
```

Пример 3. Пусть фортран-программа состоит из одной основной программы и двух подпрограмм, которые нанесены на перфокарты. Эту программу нужно оттранслировать, отредактировать и поместить в ранее созданную библиотеку загрузочных модулей BIB1 под именем PROG2. Эта библиотека размещается на магнитном диске типа 5050 с серийным номером D222. Задание на выполнение этих работ с использованием каталогизированной процедуры FORTGCL может выглядеть так:

```
//BETTA JOB
//  EXEC PROC=FORTGCL
//SYSIN DD *
    (Основная программа)
    (Первая подпрограмма)
    (Вторая подпрограмма)
/*
//LKED.SYSLMOD DD DSNAME=BIB1(PROG2),DISP=MOD,
//                    UNIT=5050,VOL=SER=D222
```

Пример 4. Пусть фортран-программа состоит из основной программы с именем MAIN и подпрограммы с именем LOG1. Эти модули оттранслированы и помещены в различные наборы данных последовательного доступа: модуль MAIN в набор B1, модуль LOG1 в набор B2. Эти наборы размещаются на магнитном диске типа 5050 с серийным номером SER5. Программа использует следующие наборы данных: входной перфокарточный набор со ссылочным номером 5 и выходной набор на бумажной ленте со ссылочным номером 6, которые имеют стандартные характеристики. Эту программу нужно оттранслировать, отредактировать и выполнить. Задание на выполнение этих работ с использованием каталогизированной процедуры FORTGLG может выглядеть так:

```
//OMEGA JOB
//  EXEC PROC=FORTGLG
//SYSIN DD  DSN=B1,DISP=SHR,UNIT=5050,
//          VOL=SER=SER5
//          DD DSN=B2,DISP=SHR,UNIT=5050,
//          VOL=SER=SER5
//GO.SYSIN DD *
    (Входные данные рабочей программы)
/*
//
```

ОТВЕТЫ И РЕШЕНИЯ

1.4.1

5. 1)—1024, 2) 8000, 3) 0, 4) 4096, 5) —0, 6) 00647, 7) 26300, 8) 19000, 9) +0

6. 1), 5), 8); 3), 7); 2), 4);

1.4.2

10. а) 1) —0. 4) 0. 7) 0.037
2) 2.641 5) —1. 8) .04
3) 999.9 6) 0.01 9) +0.0

б) 1) —.0E0 4) 0.E2 7) .37E—1
2) 2.641E0 5) —.1E+1 8) .4E—1
3) .9999E3 6) .01E+0 9) .0E0

1) —.0D0 4) .0D0 7) 3.7D—1
2).2641D+1 5) —.1D+1 8) .4D—1
3).9999D3 6) .1D—1 9) 0.0D0

11. 1), 3), 13); 2), 5), 7), 11), 15); 4), 14);

1.4.3

5. а) 1) (0.,—3.6) 4) (0.3,.29E1) 7) (.0,29.9)
2) (0.649,.0) 5) (.699E1,—314.E1) 8) (.0,14.7)
3) (—3.7,—.100.) 6) (3.,—4.) 9) (—6.999,0.)

б) 1) (.0D0,—.36D1) 6) (.3D1,—.4D1)
2) (.649D0,.0D0) 7) (.0D0,29.9D0)
3) (—.37D1,—1.D+2) 8) (.0D0,.147D2)
4) (.3D0,.29D1) 9) (—6.999D0,0.D0)
5) (.699D1,—.314D+4)

6. 1), 3), 4), 7); 2), 5), 6);

1.4.5

5. 2), 4), 6), 8)

6. а) 1) 8H—0,64201
2) 7HABCDE—2
3) 11H(—48+3*A)*B
4) 8H6A"B'—C
5) 7H,+,—, *,
6) 6H2—A **2
7) 5H3HF—V
8) 9HA'B"C'''
9) 4H ''''
10) 4H((((

б) 1) '—0,64201'
2) 'ABCDE—2'
3) '(—48+3*A)*B'
4) '6A''''B''—C'
5) ', +, —, *,'
6) '2—A **2'
7) '3HF—V'
8) 'A"B''''C''''''
9) ''''''''''
10) '((((

7. 1) 11011001110101101100011011000001
2) 11100010100101001110001110011001101000011
3) 01000000111010010100111011000001011111101
4) 11110001111100101111100 0
5) 01100000100001101111000 0
6) 1111001101101100

1.5

4. 1), 6), 7), 8), 9)
5. а) 6; б) 63;

1.6

4. 36
5. 1) 8; 2) 6; 3) 15; 4) 12; 5) 1; 6) 9.
6. 1) A (1,1), A(2,1),A(3,1),A(1,2),A(2,2),A(3,2)
2) A(1,1),A(2,1),A(1,2),A(2,2),A(1,3),A(2,3),A(1,4),A(2,4)
3) B(1,1),B(2,1),B(1,2),B(2,2),B(1,3),B(2,3)

1.7

2. Y(X) — вещественного типа стандартной длины.
3. SQRT(X) = 6.0
6. 1), 4), 5), 6), 8).

1.8

3. 1), 2), 6), 8), 9), 10), 4).
4. 1) X**Y**Z
2) 1+X+X**2/2+X**3/6
3) 1+X/(1+X/(1+X))
4) 3*SIN(X)+4*COS(X**2)**2—1
5) (X**Y/Y**X)**(Z*(Y/X))

6) (A+B)*SIN(A)+(A—B)*COS(B)+(A—B)/(SIN(A)+COS(B))
7) (X+1)/(X—1)+3.6*(X—(SIN(X)+1)**2+X**2*(SIN(X)—1))
8) —4.3—X**(Y*(6.2—Y**2))+1
9) (A—B)/(C+B/(C+B/(C—D)))
10) EXP(X)—3*SIN(X)+EXP(X**2—1)

5. 1) A*B,A*B/A,A*B/A*A,A*B/A*A/C,
A*B/A*A/C*B,—A*B/A*A/C*B
2) Z**A,Y**Z**A,X**Y**Z**A
3) X+Y, X/(X+Y), X/(X+Y)*Z, Z+X, X/(X+Y)*Z/(Z+X),
X+X/(X+Y)*Z/(Z+X), (X—Y),(X—Y)/Z,
X+X/(X+Y)*Z/(Z+X)+(X—Y)/Z
4) —3.4, B*2, A+B*2, SIN(A+B*2), SIN(A+B*2)**3,A*
SIN(A+B*2)**3, A*SIN(A+B*2)**3/2,
—3.4+A*SIN(A+B*2)**3/2,—3.4+A*SIN(A+B*2)**3/2—X

7. 1) целого типа нестандартной длины; 2), 5) целого типа стандартной длины; 3) вещественного типа нестандартной длины; 4) комплексного типа нестандартной длины.

8. (123.0619,—1.063653)

1.9

3. 3), 2)
4. 1) .TRUE.; 2) .TRUE.; 3) .FALSE.

1.10

3. 2), 4), 5)
4. 1) (A+B)*X—B.LE.0.3—A
2) X.OR..NOT.Y.AND.(X.OR.Y).AND.Y
3) .NOT.X.AND.(A*B—C)*B+D.GT.(A+B)/C.OR.X.AND..
.NOT.(X.OR.Y)
4) 3.6—2.NE.2.AND.X.AND.Y.AND..NOT.((X.OR.Y).AND.
Z)
5) .NOT.(A+B—C)/2.LT.3.6.OR.(A—B).GT.C/3

5. 1) .NOT.Y; .NOT.Z; X.AND..NOT.Z;
.NOT.Y.OR.X.AND..NOT.Z;
2) A+3.06*C.EQ.0; .NOT.Y; X.AND..NOT.Y; A+3.06*C.EQ.
0.OR.X.AND..NOT.Y
3) X.OR.Z; NOT.Z; (X.OR.Z).AND..NOT.Z; (X.OR.Z).AND..
NOT.Z.OR.X; ((X.OR.Z).AND..NOT.Z.OR.X).AND.Z; Y.OR.
((X.OR.Z).AND..NOT.Z.OR.X).AND.Z
4) Y.AND.X2J; Y.AND.X2J.AND.S; .NOT.Y; Y.AND.X2J.
AND.S.AND..NOT.Y

5) (A—D/2)*B.GT.A; .NOT.(A—D/2)*B.GT.A;.NOT.(A—D/2)*
*B.GT.A.AND.X; .NOT.Y; .NOT.(A—D/2)*B.GT.A.AND.X.
OR..NOT.Y

6. 1) .TRUE. 2) .FALSE. 3) .TRUE. 4) .TRUE.

2.1

2. 1); 3); 4); 5); 7); 9); 10)

2.2.1

4. 1) Y=SIN(A)**2+COS(A—3.14)+1
2) X=Y/(1+Y**2/(1+3*Y**3/(1+Y/(1+Y))))
3)P(X)=((((A(6)*X+A(5))*X+A(4))*X+A(3))*X+A(2))*X+A(1)

2.2.2

3. 1) X=A.NE.B
2) X=(A+B)/A+2.3.GE.Ø.3
3) X=A/(B+C)—2*B.EQ.2*C.AND.A.GE.C
4) X=.NOT.(X.OR.Y).AND.X.OR..NOT.Y
5) X=X**2—A.GT.0.2.OR.EXP(X).LT.X+2.AND.SQRT(X—
1).LE.0

4. 1) X=.TRUE. 2) X=.TRUE.

2.2.3

4. 2), если J — переменная целого типа стандартной длины;
5), если Y — переменная целого типа стандартной длины.

2.3.1

1. К оператору, помеченному меткой 16.
2. Оператор написан правильно.
3. 1), 5)
4. Управление будет передано оператору, помеченному меткой 10, т. е. самому себе.

2.3.2

3. Управление будет передано оператору, помеченному меткой 10, т. е. самому себе.
4. 1),
2), если L — переменная целого типа стандартной длины; 5), если N — переменная целого типа стандартной длины.
5. 1) 26 2) 12 3) 6

2.3.3

3. Оператор X=L+0.6
4. 1) и 2), если J и X — переменные целого типа.
5. 1) 1, 2) 5, 3) следующему, 4) 15

2.4.1

3. 2); 3); 4); 5)
4. 1) 1; 2) 2; 3) 8

2.4.2

2. 1); 3); 4)
3. 1) 5; 2) 076; 3) 104
4. 1) IF(A—3)30,30,6
   2) IF(A+B/C)9,4,2
   3) IF(A.LE.X.AND.X.LE.B.AND.C.LE.X.AND.X.LE.
      D) GO TO 100
      IF(A.LE.X.AND.X.LE.B)GO TO 104

```
5.      IF(A.GE.B) GO TO 1
        Y = —X+0.64E02
        GO TO 2
      1 X=2*A+B—1
      2 STOP
6.      Y = Z+2
        IF(Y)1,2,2
      1 T=Y**3—0.3
        GO TO 5
      2 IF(Y—1)3,3,4
      3 T = 0.0
        GO TO 5
      4 T = Y**3+Y
      5 X = T—26.3
        STOP
```

2.5.4

4. PAUSE 'STOP'
5. Подпрограмма с именем Q.
8. 4), 5), 7), 11), 13)

2.6

9. 1)
10. 1), 2)
11. 1) 17, 2) 9
12.

```
        X=0.0
        DO 1 J = 1,6
        Y(J)=X+SIN(X)
      1 X=X+0.2
```

13.
```
      A=0.0
      DO1 K=1,40
    1 A=A+B(K)
```
14.
```
      B = 0.0
      DO 2 I=1,10
      DO 2 J=1,15
    2 B=B+A(I, J)
```

3.1.1

6. 3)

7. 1) X2(2) — одномерный массив двух элементов комплексного типа стандартной длины: X2(1) = (—6.4,17.9), X2(2)=(3.7, .96E—2); величина L может быть переменной, функцией или массивом комплексного типа нестандартной длины; величина S может быть переменной, функцией или массивом комплексного типа стандартной длины;
   2) величина I логического типа нестандартной длины имеет значение .FALSE.; J(3) — одномерный массив логического типа стандартной длины. Он состоит из 3-х элементов J(1) = .TRUE., J(2)=.TRUE. Значение третьего элемента не задано.

8. 1)
```
REAL S,Q
INTEGER*2X
```
   2)
```
REAL F, C, C1, D*8,A(3)/—3.7,16.27,—1.E3/,
     B(5)/89.3E—3,0.4,—16.977E—4,.999,1.0E—15/
```

3.1.2

4. Величина SUM комплексного типа стандартной длины.

5. 3).

6. Величины BS, D1, D2, C1, C2, S1, S2, X, S12, и функция A(X) — вещественные стандартной длины; величина K1 и массив с именем L — вещественные нестандартной длины; величины K, K2, J — целого типа стандартной длины; величина S23 — комплексного типа стандартной длины; величина C — комплексного типа нестандартной длины и величина Z — логического типа стандартной длины.

3.2

3. а) величина J и массив I — целого типа стандартной длины, все остальные величины и массивы вещественного типа стандартной длины;
   б) в массиве A количество элементов 6; в массиве FOR — 3; в массиве I — 10; в C — 6; в ALPHA — 48 и в B — 4.

4.
```
REAL*8R1,R2,Z,S1,S2,CS*4/3.4E—6/,R1*4(9)/
      3*.6E—3,24.1, 4*6.7, 15./
INTEGER*J1/6/, M/17/, P2*4(4,3),J2,V
REAL A1,B,CS2,SIN
```

3.3

1. 1), 2), 5)
2. B = A = —3.7
3. F = L(5) = 8.9
4. 24 байта или 6 слов.
5.

Величина	Относительный адрес	Величина	Относительный адрес	Величина	Относительный адрес
X	48	S (1)	32	A (1)	44
Z (1)	0	S (2)	40	A (2)	48
Z (2)	16	S (3)	48	A (3)	52
Z (3)	32	S (4)	56	A (4)	56
Z (4)	48	.	.	.	.
		.	.	.	.
		.	.	.	.
		S (22)	200	A (17)	108

3.4

7. 1) неименованная область (в нее войдут величины X,Y,Z,C, X1,X2); с именем W (в нее войдут A,B,C1,C2); область с именем S (в нее войдут Q,R,T)
   2) неименованная область (в нее войдут X,Z,T,T1,T2,F,E,J); с именем R1 (в нее войдут A,B,C,S,A1,A2); с именем R2 (в нее войдут X1,X2,K,L,M).
8. 1), 2)
9. 1)
```
COMMON A(50), C(10), Z, /K/ B(7,9), X, Y, P, Q
```
   2)
```
COMMON A(50)
EQUIVALENCE (A(1), B(1), C(1))
```
   3)
```
COMMON /K/ A (50), B(7,9), Z
EQUIVALENCE (X, A(5)), (Z, Y)
```

3.5

7. 3), 4) 8. —35.65

3.7

4. Z1=1; Z2=0.0

5.
```
      REAL X(10)
      Z=-1
      DO 6 I = 1,10
      Z = Z+0.5
      Y = Z+2
      IF(Y)1,2,2
    1 T = Y**2-0.3
        GO TO 5
    2 IF(Y-1)3,3,4
    3 T = 0.0
        GO TO 5
    4 T = Y**2+Y
    5 X(I) = T+0.6E-3
    6 CONTINUE
        STOP
        END
```

6.
```
      LOGICAL D
      F(X, Y) = (X - X0)**2+(Y-Y0)**2
      Z = F (X, Y)
      IF (Z-R**2)1,1,4
    1 D=.TRUE.
      GO TO 3
    4 D=.FALSE.
    3 STOP
      END
```

4.1.1

7. Имя функции: Q; тип и длина функции: вещественный нестандартной длины. Формальные параметры: простая переменная N — целого типа стандартной длины; одномерный массив A — вещественного типа, стандартной длины, максимальное число членов массива равно N.

4.1.2

3. Первый формальный параметр — имя массива или переменной, второй и третий имена — переменных, значения которых в подпрограмме не должны вычисляться.

8. N = 15

4.1.4

7. Z = 3.0

8.

Основная программа

```
      DIMENSION F(10)
      X=—0.6
      DO 3 J=1, 10
      A=Y(X)
      IF(X)1,2,2
    1 F(J)=A+1
      GO TO 3
    2 F(J)=Z(A)**2—1
    3 X=X+0.37
      STOP
      END
```

Подпрограмма-функция

```
      FUNCTION Y(X)
      Y=X**3—X**2+X—0.4
      RETURN
      ENTRY Z(Y)
      Z=Y/(1+Y/(1+Y))
      RETURN
      END
```

4.2.3

3.

A(1,1)=A(3,2)=A(1,3)=A(1,4)=A(2,1)=A(2,3)=A(2,4)=
A(3,1)=A(3,3)=A(3,4)=A(4,1)=A(4,2)=A(4,3)=A(4,4)=0.0;
A(2,2)=A(1,2)=0.5

4. X=10.0,Y=9.0

4.2.4

4.

Основная программа

```
      REAL Y(6),X(6)/—3.43,
     2.72,9.99, 15.16,—3.99,88.8/
      DO 1 I=1,6
    1 CALL A(X(I),Y(I))
      STOP
      END
```

Подпрограмма-процедура

```
      SUBROUTINE A(X,Y)
      ‹последовательность операторов›
      RETURN
      END
```

5.

Основная программа

```
      REAL P(20)/—0.36,17.64,
        2.07,5*1.2,—3.8,
        6*—64.2,5*—1.0/
      CALL SUM(P,20,D, &2)
    2 STOP
      END
```

Подпрограмма-процедура

```
      SUBROUTINE SUM(A, N,
      B, *)
      DIMENSION A(N)
      B=0.0
      DO 1 K=1, N
    1 B=B+A(K)
      RETURN 1
      END
```

6. Основная программа

```
      DIMENSION A(10)/10*1.0/,
     X(6)/—1.6, 2.3, —16.2, 3.7,
     3.8,—4.5/, F(6)
      DO 3 K=1,6
      IF (X(K))1,2,2
    1 CALL P(A,X(K),Y,10)
      GO TO 3
    2 CALL P1(A,X(K),Y,10)
    3 F(K)=Y
      STOP
      END
```

Подпрограмма-процедура

```
      SUBROUTINE P(A,X,Y,N)
      DIMENSION A(N)
      Y=A(1)
      DO 1 J=2,N
    1 Y=Y*X+A(J)
      RETURN
      ENTRY P1 (A,X,Y,N)
      Y=0.0
      DO 2 K=1,N
    2 Y=Y+A(K)
      Y=Y+X—1.3
      RETURN
      END
```

5.1.4

3. 12 блоков.
4. 13 блоков.
5. 196 байта.
6. Длина блока — 1620 байтов, длина физической записи — 404 байта.

5.2

1. 1) A(1,1),A(2,1),A(3,1),A(4,1),A(1,2),A(2,2),A(3,2),A(4,2),A(1,3), A(2,3),A(3,3),A(4,3),A(1,4),A(2,4),A(3,4),A(4,4)
   2) A(1,1),A(1,2),A(1,3),A(1,4),A(2,1),A(2,2),A(2,3),A(2,4),A(3,1), A(3,2),A(3,3),A(3,4),A(4,1),A(4,2),A(4,3),A(4,4)
   3) A (1,2),A(2,2),A(3,2),A(1,3),A(2,3),A(3,3),A(1,4),A(2,4),A(3,4)
2. A,B(2),B(4),B(6),B(8),B(10),C(1,1),C(2,1),C(1,2),C(2,2)
3. (B(K),C(K),K=2,8,3)

6.1.1

1. а) DEFINE FILE 4(50,300,U,K)
   б) DEFINE FILE 5(1200,1210,L,K)
2. б), в)

6.1.2

1.
```
      DIMENSION XM(100)/100*1.75/, XY(200)/200*2.75/
      DEFINE FILE 2(3,100,U,K)
      WRITE (2'1)XM,XY
      RETURN
      END
```

```
2.    INTEGER S1(8,10)/80*5/
      DEFINE FILE 10(8,10,U,K)
      WRITE(10'1) ((S1(I,J),J=1,10),I=1,8)
      RETURN
      END
3.    REAL *8A,B,C,D,MAS(50)
      DEFINE FILE 3(11,15,U,J1)
      WRITE (3'1) (MAS(K),K=1,5)
      WRITE (3'2)A,B,C,D
      DO 1 I=6,46,5
      M=I+4
    1 WRITE (3'J1) (MAS(K),K=I,M)
      RETURN
      END
4.    DIMENSION A(54),B(54),C(54)
      DEFINE FILE 10(7,60,U,L)
      L= 3
      DO 1 I = 1,37,18
      M = I+17
    1 WRITE (10'L) (A(K), B(K), C(K), K = I, M)
      RETURN
      END
5.    DEFINE FILE 9(50, 51, L, K)
      INTEGER *2 MAC1 (250)
      DO 1 I = 1,250
    1 MAC1(I) = MAC1(I)*I
      WRITE (9'10) MAC1
      RETURN
      END
```

6.1.3

1. A=21.01, Q = 6.13, B = −3.21,R=0.1, S=6.15
2. A(1)=−64.1,A(2)=11.2,A(3)=9.6,A(4)=0.3,A(5)=12.0, A(6)=6.4,A(7)=18.1,A(8)=A(9)=8.1,A(10)=−17.01
3. X(1,1)=−0.17,X(1,2)=−6.18,X(1,3)=0.28,X(1,4)= =−0.501E03,X(1,5)=6.13,X(2,1)=1.02,X(2,2)=0.28, X(2,3)=−709.8,X(2,4)=−0.501E03,X(2,5)=6.18,X(3,1)= = −6.18,X(3,2)=0.28,X(3,3)=29.302,X(3,4)=2.18E− 1,X(3,5)=11.16,X(4,1)=−6.18,X(4,2)=0.28,X(4,3)= =−9.34,X(4,4)=6.13,X(4,5)=8.01
5. DEFINE FILE 2(3,6,U,L)

```
      DIMENSION MW(15)/11,12,13,14,15,21,22,23,24,25,31,32,33,
      34,35/,M1(7),M2(5),M3(3)
      L=1
      DO 1 I=1,11,5
      M=I+4
    1 WRITE (2'L) (MW(J), J = I, M)
      READ (2'1)(M1(I),I=1,5)
      READ (2'2)(M1(I),I=6,7),(M2(K),K=1,3)
      READ (2'3)(M2(I),I=4,5),M3
      RETURN
      END
6.    REAL MATR(15,10)
      DEFINE FILE 9(30,10,U,K)
      READ (9'16)((MATR(I,J),J=1,10),I=1,15)
      RETURN
      END
7.    DEFINE FILE 15(22,20,U,L)
      REAL B(20,20),A(20),C(20)
      L=1
      READ(15'L)A,((B(I,J),J=1,20),I=1,20)
      DO 1 I=1,20
      C(I)=0
      DO I J=1,20
    1 C(I)=C(I)+B(I,J)*A(J)
      WRITE (15'22)C
      RETURN
      END
8.    DEFINE FILE 10(15,101,L,K),11(15,101,L,K1)
      LOGICAL *1M(101)
      DO 1 I=1,15
      READ (10'I)M
    1 WRITE(11'I)M
      RETURN
      END
```

6.2.1

```
1.    DEFINE FILE 17(2,50,U,K)
      DIMENSION M(100)
      READ (17'1)M
      WRITE (16)M
      RETURN
      END
```

После выполнения программы набор данных 16 будет содержать три блока; первые два блока будут иметь максимальную длину равную 188 байтам, третий блок будет иметь длину равную 48 байтам.

2.
```
REAL*8RM1(100),RM2(10,5),RM3(7,10)
WRITE (2)RM1
WRITE (2)RM2, RM3
RETURN
END
```
Набор данных 2 содержит три блока, длина первого блока — 608 байтов, длина второго блока — 612 байтов, длина третьего блока — 568 байтов.

3.
```
DEFINE FILE 9(10,60,U,K)
REAL MS1(50)
READ (9'5)MS1
DO 10 I =1,10
10 MS1(I)=MS1(I)**2+0.5
WRITE (1)(MS1(I),I=1,10)
WRITE (1)(MS1(I),I=11,50)
RETURN
END
```
Набор данных 1 состоит из пяти блоков, все блоки имеют одинаковую длину, равную 48 байтам.

6.2.2

1. X(1)=22.7, X(2)=6.34, X(3)=15.2, X(4)=—4.8, X(5)=6.8, X(6)=—9.18, Z=—38.01, Y=107.07
2. X(1)=20, X(2)=—212, X(3)=6, X(4)=16, X(5)=6, X(6)=—5, X(7)=10, X(8)=3, X(9)=8, X(10)=8
3. X(1,1)=0.0, X(1,2)=0.0, X(1,3)=0.0, X(1,4)=0.0, X(1,5)=0.0, X(2,1)=0.0, X(2,2)=—276.3, X(2,3)=0.0, X(2,4)=0.0, X(2,5)=32.8, X(3,1)=0.0, X(3,2)=99.5, X(3,3)=0.0, X(3,4)=0.0, X(3,5)=108.9, X(4,1)=0.0, X(4,2)=68.10, X(4,3)=0.0, X(4,4)=0.0, X(4,5)=64.27, X(5,1)=0.0, X(5,2)=18.03, X(5,3)=0.0, X(5,4)=0.0, X(5,5)=0.1, A=—0.02
4. Первая запись содержит числа: 2,—4,5,—5,3; вторая запись — число 32.
5. Первая запись содержит числа: 12.1,0.5,—4.1,6.2,—4.12,10; вторая запись — числа: —0.12,14.0,0.5,0.5,5.0,6.2,6.2,—4.12,9.1.

```
6.    INTEGER A(10)
      WRITE (23)(A(I),I=2,10,2)
      WRITE (23)(A(I),I=1,9,2)
      WRITE (23)(A(I),A(I+1),I=1,7,3)
      RETURN
      END
7.    INTEGER M1(35),M2(35)
      READ (2) (M2(35—(I—1)),I=1,35)
      RETURN
      END
8.    INTEGER M1(30),M2(40)
      READ (3)
      READ (3)(M2(I),I=11,40)
      RETURN
      END
9.    REAL X(50),Y(25)
      READ (5,ERR=2)X
      DO 1 I=1,25
    1 Y(I)=X(I+25)
      RETURN
    2 STOP 3
      END
10.   DEFINE FILE 2(50,150,L,K)
      REAL *8 A(15)
      K=1
    1 READ (1,END=2)A
      WRITE (2'K)A
      GO TO 1
    2 RETURN
      END
```

7.1

4. I4 (для переменной I), I3 (для переменной J), I3 (для переменной K), I4 (для переменной M).
5. Форма представления значения каждого элемента массива M1 описывается форматом I8.
6. От —999 до 9999.

7.2

1. От —999.99 до 9999.99
2. С точностью до 3-х знаков после десятичной точки.
3. F8.5

4. F10.4 (для вещественных и мнимых компонент переменных A и B); F12.4 (для вещественной компоненты переменной C); F8.4 (для мнимой компоненты переменной C).

7.3

6. а) _—0.7615E_02__0.3001E_01_—0.9900E—01
   б) _____—76.1500_____3.0010_____—0.0990
7. E13.4;E13.3;E12.4.

7.4

5. _ —0.1643D_07_—0.1900D_06__0.9901D_02__ 0.3699D_02_—0.9201D_03
6. Количество переменных равно 3. Форматы переменных: D12.3, D11.3, D10.2

7.6

1. а) __T (основная форма), T__,TTT,ATB,AAT
   б) __F (основная форма), F15,F__,—F_,ABF
3. L4

7.7

2. XYZW___*1234
3. а) A=APTA,б)A=APTA,в)A=8MAPTA__

7.8

2. а) CD,б)1234EFCD,в)__1234EFCD
3. а) 01ACDF00,б)DF00,в)0000000001ACDF00

7.12

3. A(1) будет иметь вид _0. 159E—01,
   A(2) будет иметь вид __1.59____,
   A(3) будет иметь вид __15.9____,
   A(4) будет иметь вид __159.____,
   A(5) будет иметь вид _0.159E_04
4. Переменная A будет иметь вид __0.5555E—01
   переменная K будет иметь вид _____—21
   переменная L будет иметь вид _____F
5. а) тип — целый, длина — стандартная или нестандартная, формат — I4 или G4; б) тип — вещественный, длина — стандартная или нестандартная, формат — F8.3 или G8.3; в) тип — вещественный, длина—стандартная, формат—E11.3 или G11.3; г) тип вещественный, длина—нестандартная, формат D10.3 или G10.3

7.13

1. 3E7.2
2. I5,L4,L4,I5,L4,L4,I5,L4,L4
3. а) может, б) может, в) не может, г) не может.
4. 10 логических записей.
5. 4 логические записи.
6. а) I5, б) 3(I6), в) 2(I7,I4), г) 2(I7,3(I4))
7. Всегда.
8. Не всегда.

7.14

3. а) ␣␣14.385, б) 1438.500 в) ␣␣0.14385E␣02,
   г) ␣␣0.00014E␣05, д) ␣␣␣14.385␣␣␣␣␣,
   е) ␣␣␣14.385␣␣␣␣␣,
4. а) —31105.007, б) —3110.5007, в) —311050.07, г) —31105.007,
   д) —31105.007, е) —311050.07

8.1.1

3. ␣␣␣␣␣357␣␣␣␣␣—476␣␣␣␣2401␣␣␣␣␣␣
4. В десятую запись будут помещены символы:
   ␣␣␣␣␣0.218E␣01␣␣␣␣␣0.418E␣01␣␣␣␣␣
   0.618E␣01␣␣␣␣␣0.818E␣01␣␣␣␣␣0.102E␣02␣
   ␣␣␣␣0.122E␣02␣
   В одиннадцатую запись будут помещены символы:
   ␣␣␣␣␣0.142E␣02␣␣␣␣␣0.162E␣02␣␣␣␣␣
   0.182E␣02␣␣␣␣␣0.202E␣02
   и двадцать девять символов «пробел».
5. В десятую запись будут помещены символы:
   ␣␣␣␣␣0.218E␣01␣␣␣␣␣0.418E␣01␣␣␣␣␣
   0.618E␣01␣␣␣␣␣0.818E␣01␣␣␣␣␣0.102E␣02
   и пятнадцать пробелов.
   В одиннадцатую запись будут помещены символы:
   ␣␣␣␣␣0.122E␣02␣␣␣␣␣0.142E␣02␣␣␣␣␣
   0.162E␣02␣␣␣␣␣0.182E␣02␣␣␣␣␣0.202E␣02
   и пятнадцать пробелов.
6. ПЕТРОВ, ИВАНОВ␣␣
7. В третью запись будут помещены символы:—2.3017.08␣␣␣␣␣
   В четвертую запись будут помещены символы: ␣123␣—35␣␣
   ␣␣␣␣␣
   В пятую запись будут помещены символы: ␣F и тринадцать пробелов.

8. Четвертая запись: _ _ _ _ _ 11_11_11 и двадцать семь пробелов. Пятая запись: _ _ _ _ —135 и тридцать два пробела.

9. а)
```
      DEFINE FILE 1(15,3,U,K)
       INTEGER A, B, C
      DEFINE FILE 2(4,50,E,L)
    1 FORMAT (3I5)
      READ (1'1)A,B,C
      WRITE (2'2,1)A,B,C
      RETURN
      END
```
б)
```
      DEFINE FILE 1(15,3,U,K)
      INTEGER A,B,C
      DEFINE FILE 2(4,50,E,L)
    1 FORMAT (10('*'), 3I5,25('*'))
      READ (1'1)A,B,C
      WRITE (2'2,1)A,B,C
      RETURN
      END
```

10.
```
      DEFINE FILE 1(5,85,E,L)
      REAL A(10)/10*3.5/,B(10)/10*7.8/,C(10)
    1 FORMAT ('C=',9(F7.5,1H,),F7.5)
      DO 2 I=1,10
    2 C(I)=A(I)*B(I)
      WRITE (1'1,1)C
      RETURN
      END
```

8.1.2

3. I=—140, X=.TRUE.

4. а)
```
      DEFINE FILE 1(20,48,E,L)
      DEFINE FILE 2(10,32,E,K)
      DIMENSION M(8)
    1 FORMAT (6I6/2I6)
    2 FORMAT (8I4)
      L=1
      K=1
      DO 3 I=1,10
      READ (1'L,1)M
    3 WRITE (2'K,2)M
      RETURN
      END
```
б)
```
      DEFINE FILE 1(20,48,E,L)
```

```
      DEFINE FILE 2(10,32,E,K)
      DIMENSION M(8)
    1 FORMAT (6A6/2A6)
    2 FORMAT (8A4)
      L=1
      K=1
      DO 3 I=1,10
      READ (1'L,1)M
    3 WRITE (2'K,2)M
      RETURN
      END
```

5. `I = −246,X=3620.0,A=0.623,B=5.27,C=0.0222,R=24.5`

6.
```
      DEFINE FILE 20(15,50,E,K)
    1 FORMAT (10I5)
    2 FORMAT (2I5,(4E10.3))
      DIMENSION M(12),X(8)
      READ (20'2,1) (M(I),I=3,12)
      READ (20'10,2)M(1),M(2),X
      RETURN
      END
```

8.2.1

2. `_F_T___−0.34____4.03_−10.05_−10.05_−10.05_−10.05_−10.05`

3. 1) 4 блока; 2) блоки 1, 2 и 3 — по две записи, блок 4 — одну запись; 3) в 1-ю запись будут помещены `_МАТРИЦА_А` и сорок шесть символов «пробел»; во 2-ю запись — пятьдесят шесть символов «пробел», в 3-ю запись — символы `_0.40`, девять последовательностей символов `_0.25` и шесть символов «пробел», в 4-ю запись — десять последовательностей символов `_0.30` и шесть символов «пробел», в 5-ю запись — десять последовательностей символов `_0.35` и шесть символов «пробел», в 6-ю запись — десять последовательностей символов `_0.40` и шесть символов «пробел», в 7-ю запись — десять последовательностей символов `_0.45` и шесть символов «пробел».

4. 1) 3 блока; 2) блок 1—3 записи, блоки 2 и 3 — по две записи;

5.
```
      INTEGER M (10)/−104,91,−36,27,1,6,87,95,−3,18/
      COMPLEX A/(36.5,−24.3)/,B/(7.42,−0.17)/
    1 FORMAT (4E10.3,(5I5))
      WRITE (9.1)A,B,M
      RETURN
      END
```

```
6. 1)  DEFINE FILE 1(10,120,E,K)
       REAL *8A(10)
     1 FORMAT (10D12.3)
     2 FORMAT (10F15.3)
       K=1
       DO 3 I=1,10
       READ (1'K,I)A
     3 WRITE (2,2)A
       RETURN
       END
  2)   DEFINE FILE 1(10,120.E,K)
       REAL*8A(15)
     1 FORMAT (10D12.3/5D12.3)
     2 FORMAT (15F12.3)
       K=1
       DO 3 I=1,5
       READ (1'K,1)A
     3 WRITE (2,2)A
       RETURN
       END
7.     DEFINE FILE 8(8,300,E,K)
       INTEGER A(100)
     1 FORMAT (5016///5016)
     2 FORMAT (////(T11,25(15,5H*****)))
       READ (8'1,1)A
       WRITE (9,2)A
       RETURN
       END
```

8. б) В первую запись будут помещены

_ _ F _ _ 3.5 _ _ _ _ —0.6 _ _ _ _ 0.125E _ 01 _ _ —2.5 _ _ _ _ _ 0.1 _ _ _ _ _ _ 1

Во вторую запись будут помещены

_ 0.167E _ 01 _ _ 1.7 _ _ _ _ _ 0.1 _ _ _ _ _ _ 2

В третью запись будут помещены

_ 0.158E _ 01 _ _ 0.8 _ _ _ _ _ 0.1 _ _ _ _ _ _ 3

8.2.2

1. C=(0.241E1,—0.89E—1),B=.TRUE.,M1(1)=—16,M1(2)=69, M1(3)=100, M1(4)=58, M1(5)=3,M1(6)=—119, M1(7)=2, M1(8)=—4.

2. MAT(1,1)=411, MAT(1,2)=512, MAT(1,3)=613, MAT(1,4)=714, MAT(2,1)=421, MAT(2,2)=522, MAT(2,3)=623,MAT(2,4)=724,MAT(3,1)=431,

```
      MAT(3,2)=532,MAT(3,3)=633,MAT(3,4)=734,
      K(1)=24,K(2)=64,K(3)=323,B(1)=—3.14,B(2)=5.93
3.    A1=265.4,NA(1)=11,NA(2)=12,NA(3)=13,NA(4)=21,
      NA(5)=22,NA(6)=23,A(1,1)=—27.5,A(1,2)=249.2,
      A(1.3)=—54.2,A(2,1)=—34.2,A(2,2)=—45.5,A(2,3)=0.2,
      B(1)=57.3,B(2)=—68.9,B(3)=—76.8
4.    REAL A(2,3),B(3,2)
    1 FORMAT (2(I2,X),6(F5.2,X))
      READ (12,1)K,N,((B(J,I),A(I,J),J=1,3),I=1,2)
      RETURN
      END
5.    REAL *8 A,B(50)
    1 FORMAT (T5, 50D14.5)
      A=0
    2 READ (1,1,END=4)B
      DO 3 I=1,50
    3 A=A+B(I)
      GO TO 2
    4 RETURN
      END
```

8.2.3

1.

```
МАСШТАБ␣X
␣␣␣␣␣␣␣␣␣
X1=␣1.25
X2=␣2.25
X3=␣3.25
X4=␣4.25
X5=␣5.25
```

2.

```
I␣␣␣␣␣X␣␣␣␣␣I␣␣␣␣␣Y␣␣␣␣I
I␣␣2.5000E-03␣␣I␣␣␣␣0.1␣␣I
I␣␣2.5500E-02␣␣I␣␣␣␣1.0␣␣I
I␣␣2.5500E-01␣␣I␣␣␣10.0␣␣I
I␣␣3.5300E␣00␣␣I␣␣100.0␣␣I
I␣␣2.5500E-01␣␣I␣I000.0␣␣I
```

```
3. 1  FORMAT ('1ЗНАЧЕНИЯ␣ПЕРЕМЕННЫХ␣S,R,S1,R1'/
      ('0',2G15.4))
      WRITE (6,1)S,R,S1,R1
4.  1 FORMAT (F7,3)
    2 FORMAT ('1', 'A=', E12.5,';B=',E12.5,'; '/ '␣C=',
      E12.5,';D=',E12.5,';')
      READ (5,1)A,B,C,D
```

```
      WRITE (6,2)A,B,C,D
      RETURN
      END
```

9.3.2

```
3.    //FOR1 JOB COND=(8,LT)
4.    //SYS1 JOB MSGLEVEL=1,COND=(4,EQ),TIME=25,
                CLASS=B
```

9.3.3

```
2.    // EXEC PGM=PROG1
```

9.3.4

```
3.    //FT06F001 DD DUMMY
```

9.3.5

```
3. а) //FT05F001 DD *
   б) //FT05F001 DD UNIT=183
4. а) //FT08F001 DD *
   б) //FT08F001 DD UNIT=183,DCB=(RECFM=F,
      //                BLKSIZE=80)
5. а) //FT07F001 DD *
   б) //FT07F001 DD UNIT=183
```

9.3.6

```
3.  а) //FT07F001 DD SYSOUT=B
    б) //FT07F001 DD UNIT=182
4.  а) //FT04F001 DD SYSOUT=B
    б) //FT04F001 DD UNIT=182, DCB=(RECFM=F,
       //                BLKSIZE=80)
5.  а) //FT05F001 DD SYSOUT=B
    б) //FT05F001 DD UNIT=182
```

9.3.7

```
3.  а) //FT06F001 DD SYSOUT=A
    б) //FT06F001 DD UNIT=182
5.  а) //FT06F001 DD SYSOUT=A, DCB=(RECFM=U,
       //                BLKSIZE=128)
    б) //FT06F001 DD UNIT=182, DCB=(RECFM=U,
       //                BLKSIZE=128)
```

9.3.8

```
2.    //FT03F001 DD DSN=TABL, VOL=SER=221223,
      //                DCB=(RECFM=F,BLKSIZE=128),
      //                SPACE=(CYL,15),DISP=(NEW,KEEP),
      //                UNIT=023
```

3.
```
//EXEC PGM=PROG2
//FT02F001 DD DSN=*.STEP1.FT01F001,
//            DISP=(OLD,DELETE)
```

9.3.9

4.
```
//FT38F001 DD DSN=TXP,DISP=SHR,
//            VOL=SER=L00011, LABEL=2,UNIT=5010
```

10.1

3.
```
// EXEC KP1
```

4. а)
```
// EXEC KP2
```
б)
```
// EXEC KP2, PARM.P1=2
```
в)
```
// EXEC KP2,PARM.P1=,REGION.P2=75K
```
г)
```
// EXEC KP2
//P1. D1 DD UNIT=5050
```
д)
```
// EXEC KP2,PARM.P2=A
//P1.D1 DD DISP=NEW
//P2.D3 DD SYSOUT=B
```

10.2

4.
```
// EXEC PGM=IEYFORT,PARM=(DECK,SOURCE)
//SYSPUNCH DD SYSOUT=B
//SYSPRINT DD SYSOUT=A
//SYSIN    DD *
исходная фортран-программа
/*
//
```

10.3

5.
```
// EXEC PGM=IEWL
//SYSLIB   DD DSN=SYS1.FORTLIB,DISP=SHR
//SYSLIN   DD DSN=N1,DISP=SHR,UNIT=5050,
//            VOL=SER=S12
//             DD DSN=N2,DISP=SHR,UNIT=5050,
//            VOL=SER=S12
//           DD DSN=N3,DISP=SNR,UNIT=5050,
//              VOL=SER=S12
//SYSUT1   DD UNIT=SYSDA,SPACE=(1024,20)
//SYSPRINT DD SYSOUT=A
//SYSLMOD  DD DSN=B1(PROG5), DISP=MOD,
//            UNIT=5050,VOL=SER=S12
//
```