З.С.БРИЧ
Д.В.КАПИЛЕВИЧ
О.Г.ТЕРЕХОВА

Программирование на ФОРТРАНЕ ЕС ЭВМ в режиме разделения времени

Москва «Финансы и статистика» 1982

ББК 32.973
Б87

Зинаида Сергеевна Брич
Дора Вениаминовна Капилевич
Ольга Геннадьевна Терехова

ПРОГРАММИРОВАНИЕ НА ФОРТРАНЕ ЕС ЭВМ В РЕЖИМЕ РАЗДЕЛЕНИЯ ВРЕМЕНИ

Рецензент **А. Л. Александров**
Зав. редакцией **И. Г. Дмитриева**
Редактор **Л. Д. Григорьева**
Мл. редакторы **В. К. Капинская, Г. В. Розанова**
Техн. редакторы **К. К. Букалова, Л. Г. Челышева**
Корректоры **Г. А. Башарина, Г. И. Терновская**
Худож. редактор **О. Н. Поленова**
Обложка художника **А. М. Ясинского**

ИБ № 1089

Сдано в набор 30.11.81. Подписано в печать 09.04.82. А04295. Формат 84×108$^{1}/_{32}$. Бум. тип. № 2. Гарнитура «Литературная». Печать высокая. П. л. 6,0. Уч.-изд. л. 10,42. Усл. п. л. 10,08. Усл. кр.-отт. 10,29. Тираж 20 000 экз. Заказ № 8662. Цена 55 коп.

Издательство «Финансы и статистика», Москва, ул. Чернышевского, 7.

Областная типография управления издательств, полиграфии и книжной торговли Ивановского облисполкома, 153628, г. Иваново, ул. Типографская, 6.

Брич З. С. и др.

Б87 Программирование на Фортране ЕС ЭВМ в режиме разделения времени/З. С. Брич, Д. В. Капилевич, О. Г. Терехова. — М.: Финансы и статистика, 1982. — 192 с., ил.

55 коп.

Описана система программирования Фортран для режима разделения времени операционной системы ОС ЕС. Рассматриваются средства СРВ для создания и трансляции программ на Фортране. Даны рекомендации по использованию новых трансляторов с языка Фортран в режиме разделения времени и в режиме пакетной обработки. Большое внимание уделяется описанию возможностей отладки программ на Фортране в диалоговом режиме.

Для широкого круга программистов, работающих с языком Фортран.

Б $\frac{2405000000-063}{010(01)-82}$ 116—82

ББК 32.973
6Ф7.3

ПРЕДИСЛОВИЕ

Новые режимы операционной системы ОС ЕС ЭВМ определили направление развития систем программирования с широко распространенных алгоритмических языков Фортран, Кобол и ПЛ/1. Появление режима разделения времени, который предоставляет простые и удобные диалоговые средства общения с ЭВМ, привело к созданию в системах программирования посредников по вызову трансляторов в среде разделения времени и диалоговых средств ведения отладки на входном языке.

Пользователю системы программирования, работающему в среде разделения времени, доступны все возможности, предоставляемые системой разделения времени (СРВ). По сравнению с пакетной обработкой она имеет ряд преимуществ по использованию ЭВМ. При работе в режиме пакетной обработки составленное пользователем задание на трансляцию, редактирование и выполнение программы поступает оператору ЭВМ, который включает его в пакет заданий. Сформированный пакет заданий оператор запускает для обработки на ЭВМ без участия пользователя. Отсутствие прямых контактов пользователя с ЭВМ в режиме пакетной обработки не позволяет ему сразу исправлять свои ошибки. Пользователю требуется за рабочим столом проанализировать результаты обработки своего задания и подготовить новый вариант задания, что замедляет отладку и увеличивает время получения окончательных результатов.

В отличие от пакетной обработки режим разделения времени предоставляет пользователю необходимые ему ресурсы вычислительной системы на длительное время. Используя средства СРВ и системы программирования, пользователь во время сеанса работы за абонентским пунктом (АП) может подготовить синтаксически правильную программу, протранслировать, отредактировать ее и приступить к отладке. Средства от-

ладки позволяют в режиме диалога провести контроль за выполнением программы на уровне входного языка. Степень готовности программы за одноразовое использование абонентского пункта в этом случае фактически зависит от подготовленности пользователя к работе в системе разделения времени и от сложности разрабатываемой программы.

Система разделения времени делает доступными ресурсы ЭВМ одновременно нескольким пользователям. Количество параллельно работающих пользователей зависит от числа абонентских пунктов, обслуживаемых системой разделения времени.

Системы программирования не накладывают ограничения на типы абонентских пунктов. Пользователь системы программирования может вести обработку своей программы с любого типа абонентского пункта, который поддерживается системой разделения времени. Так, система разделения времени MVT издания 1.2 или SVS издания 1.0 операционной системы ОС издания 6.1 позволяет работать за абонентскими пунктами типа ЕС-7906, ЕС-7920, ЕС-8570, ЕС-8561, ЕС-8562, ЕС-8563, ЕС-8564. Совершенствование системы разделения времени в части обеспечения новых типов абонентских пунктов автоматически распространяется и на системы программирования.

Система программирования Фортран, описываемая в данной книге, предоставляет пользователю средства разработки программы на Фортране в диалоговом режиме и в то же время обеспечивает его возможностями для выполнения заданий в пакетном режиме. Книга содержит сведения, необходимые пользователю для эксплуатации этой системы.

Авторы сочли возможным не включать в книгу описание языка Фортран Единой Системы ЭВМ и сослаться на [1] с оговоркой, что в настоящей книге используется терминология, принятая в ГОСТ 23056—78 «Язык программирования Фортран».

Авторы благодарят своих коллег по разработке системы программирования Фортран за полезные замечания к тексту книги.

Раздел I

ПРОГРАММИРОВАНИЕ В СРВ

Общие сведения

Глава 1

СОСТАВ СИСТЕМЫ ПРОГРАММИРОВАНИЯ ФОРТРАН

1.1. ОБЩИЕ СВЕДЕНИЯ О СИСТЕМЕ ПРОГРАММИРОВАНИЯ ФОРТРАН ДЛЯ СРВ

Язык программирования Фортран получил широкое распространение среди пользователей ЕС ЭВМ. Он компактен, прост для освоения и особенно удобен для программирования задач научно-технического характера, в которых преобладают математические вычисления. Для разработки программ на Фортране в среде разделения времени предназначается пакет прикладных программ под названием «Система программирования Фортран для режима разделения времени операционной системы ОС ЕС». Пакет включает ряд компонентов, которые вместе со стандартными средствами СРВ обеспечивают полный цикл обработки программ на Фортране (трансляцию, редактирование, отладку и выполнение) в диалоговом режиме. Система программирования Фортран в среде разделения времени позволяет:

вводить исходную программу непосредственно с АП, тем самым исключая подготовку ее на таком традиционном носителе данных, как перфокарты;

подготавливать предложения исходной программы в свободном формате, позволяющем размещать метку и текст предложения с любой позиции строки;

выполнять предварительную проверку синтаксиса программы при ее вводе и в случае обнаружения ошибок вносить необходимые корректировки;

использовать один из двух имеющихся трансляторов в зависимости от сложности решаемой задачи, квалификации пользователя, срочности получения результатов трансляции или результатов счета и требований к содержанию выходной информации трансляторов;

обращаться к транслятору с помощью одной команды СРВ, обеспечивающей установку режимов и распределение необходимых для транслятора наборов данных;

производить диалоговую отладку рабочей программы на Фортране на уровне входного языка;

корректировать исходную программу по результатам отладки;

выполнять полный цикл обработки, заканчивающийся получением результатов счета.

Эти возможности системы программирования Фортран легко доступны пользователям благодаря простым средствам общения с ЭВМ, предоставляемым им в СРВ. Эта система может быть без затруднений освоена инженерами, математиками, студентами и другими пользователями, которые знакомы с языком Фортран, но не являются профессиональными программистами.

Система программирования Фортран для СРВ содержит следующие компоненты: трансляторы Фортран SE и Фортран CC, посредники для вызова этих трансляторов, программу преобразования формата исходной программы, библиотеку Фортрана, Диалоговый отладчик. Система Фортран для СРВ полностью реализует возможности языка, изложенные в [1]. В системе имеются расширения языка, касающиеся ввода-вывода (см. 1.2).

1.1.1. Транслятор Фортран SE

Транслятор Фортран SE является расширенной версией транслятора Фортран ST, который входит в состав операционной системы ОС. Так как транслятор Фортран SE выполняется и в пакетном режиме, он может заменить транслятор Фортран ST, который предназначен для выполнения только в пакетном режиме. Объектные модули трансляторов Фортран ST и Фортран SE совместимы. Поэтому программы, полученные

в результате эксплуатации транслятора Фортран ST, могут использоваться при разработке последующих задач с помощью трансляторов этой системы. Наиболее важные отличия транслятора Фортран SE от транслятора Фортран ST: расширение средств ввода-вывода языка Фортран, улучшение алгоритмов перевода данных, настройка транслятора Фортран SE на использование в режиме разделения времени.

Транслятор Фортран SE производит оптимизацию циклов, поэтому выдаваемый им объектный код эффективнее, чем объектный код, построенный транслятором Фортран CC. Транслятор Фортран SE предназначается пользователям, длительно эксплуатирующим свои программы, для которых существенное значение имеет эффективность объектного кода. Пользователь может запросить такие режимы трансляции, которые позволяют ему сохранить объектный модуль в наборе данных, а также получить для документирования распечатки исходной программы, распределения памяти и текста объектного модуля. Для оперативной работы за абонентским пунктом в режиме разделения времени можно запросить режим трансляции, по которому производится вывод на абонентский пункт только ошибочных предложений и сообщений о допущенных в них ошибках. Для проведения диалоговой отладки трансляцию необходимо выполнить со специальным режимом.

1.1.2. Транслятор Фортран CC

Транслятор Фортран CC предназначен для пользователей, которые решают сравнительно небольшие задачи разового характера. Он может выполняться в пакетном режиме и в режиме разделения времени. Особенно полезен этот транслятор для обучения языку Фортран. Так как начинающим изучение языка Фортран приходится многократно выполнять трансляцию, то в этом случае важны удобство в использовании и скорость трансляции, а не эффективность создаваемого объектного кода. Эти качества достигнуты включением в транслятор режима обработки предложений Фортрана в свободном формате, расширенной диагностики ошибок в исходной программе, минимального вывода результатов трансляции на АП (только текста сообще-

ний об ошибках), выполнения полного цикла обработки программы за одно обращение к транслятору, сохранения объектной программы в основной памяти.

В случае успешной трансляции транслятор вызывает Загрузчик операционной системы для редактирования объектного модуля, подготовленного в основной памяти. Используя библиотеку Фортрана, Загрузчик формирует рабочую программу и передает ей управление. Таким образом, синтаксически правильная программа за одно обращение к транслятору проходит полный цикл обработки, заканчивающийся выполнением программы.

Благодаря простоте в использовании транслятор Фортран СС особенно удобен для неквалифицированного пользователя, так как позволяет ему минимально обращаться к средствам операционной системы. Квалифицированному пользователю этот транслятор может быть полезен, например, при отработке некоторых алгоритмов отдельных частей большой задачи. Когда программа отлажена и в набор данных с исходной программой внесены все корректировки, пользователь может обратиться к транслятору Фортран SE для получения более эффективного объектного кода и сохранения результатов трансляции для их последующего использования.

Как и для транслятора Фортран SE, последующая диалоговая отладка программы обеспечивается по специальному режиму трансляции.

1.1.3. Посредники для вызова трансляторов

Интерфейсом между системой разделения времени и трансляторами являются посредники. Посредник позволяет обращаться к транслятору с помощью одной команды СРВ, в которой указываются имя набора данных с исходной программой и режимы выполнения транслятора. Посредник обеспечивает динамическое распределение необходимых транслятору наборов данных в зависимости от режимов трансляции, подготовку списка режимов и вызов транслятора. Если абонент при вводе команды для вызова транслятора допускает ошибку, посредник выдает подсказывающее сообщение.

1.1.4. Программа преобразования формата предложений исходной программы

Транслятор Фортран СС может обрабатывать исходные программы, содержащие предложения Фортрана в стандартном или свободном формате. Транслятор Фортран SE допускает только стандартный формат предложений. Для преобразования формата предложений Фортрана предназначена программа преобразования CONVERT. К ней обращаются, например, когда программа, подготовленная в свободном формате, должна быть обработана транслятором Фортран SE или в программу, записанную в стандартном формате, вносятся изменения в свободном формате. Программа CONVERT может выполняться как в пакетном режиме, так и в режиме разделения времени.

1.1.5. Библиотека Фортрана

Для реализации стандартных математических функций и операций ввода-вывода используется библиотека Фортрана. Эта библиотека представляет улучшенную версию библиотеки Фортрана ОС ЕС, так как в ней использованы алгоритмы более точного перевода данных и включены дополнительные программы, обеспечивающие новые возможности ввода-вывода. Поэтому рекомендуется предыдущую версию библиотеки заменить библиотекой системы программирования Фортран.

1.1.6. Диалоговый отладчик

Диалоговый отладчик используется для отладки рабочей программы, протранслированной в специальном режиме. Задание отладочных действий производится с помощью языка диалоговой отладки, состоящего из набора подкоманд команды CPB. В подкомандах указываются объекты исходной программы, над которыми выполняются отладочные действия. Эти действия распространяются на такие объекты исходной программы, как переменные, массивы, метки и номера строк операторов. Отладочные действия выполняются в точках прерывания, которые задаются метками или номерами строк операторов исходной программы. Точка преры-

вания указывает оператор, перед которым необходимо остановить выполнение программы.

После приостановки рабочей программы в заданных точках прерывания пользователь может выполнить следующие действия: распечатать значения переменных и массивов; присвоить переменным и массивам новые значения; установить режимы обработки ошибок, которые могут возникнуть при выполнении программы; распечатать текст предложений исходной программы; задать режим слежения за ходом выполнения программы; задать новые точки прерывания или отменить те, которые уже не требуются; задать выполнение сервисных функций (например, вывод сведений об активных точках прерывания).

После выполнения отладочных действий в точке прерывания пользователь может продолжить ход рабочей программы, предусмотренный алгоритмом, либо изменить его.

1.2. НОВЫЕ ВОЗМОЖНОСТИ ЯЗЫКА ФОРТРАН

Язык Фортран, реализованный в системе программирования Фортран для СРВ, по сравнению с предыдущей версией языка Фортран ОС ЕС расширен операторами ввода-вывода, управляемого списком. Эти операторы позволяют, не кодируя объявление FORMAT, вводить и выводить данные по формату в соответствии с их типом.

Общая форма операторов ввода-вывода:

```
READ(a,*)список
WRITE(a,*)список
```

где a — номер набора данных; * — признак ввода-вывода, управляемого списком; список — список ввода-вывода.

Символ «*» записывается вместо метки объявления FORMAT.

Правила подготовки данных для ввода, управляемого списком, не такие жесткие, как при вводе по формату, заданному объявлением FORMAT. Каждое данное во внешнем представлении занимает произвольное количество позиций и кодируется в любом формате, допустимом для ввода данных. Тип данного должен согласовываться с типом соответствующего ему элемента в списке ввода. Исключение составляют текстовые дан-

ные, которые могут указываться для переменных любого типа. Для текстовых данных допускается только одна форма представления — в виде литерала, заключенного в апострофы. Числовые данные не должны содержать внутренних пробелов. Знак «+» и десятичная точка необязательны. Если в вещественной константе отсутствует десятичная точка, то она подразумевается за самой правой цифрой константы. Вещественная константа может содержать экспоненту с буквой E или D. Комплексное данное представляется в виде упорядоченной пары вещественных констант, разделенных запятой и заключенных в круглые скобки. Логическое данное содержит букву T (истина) или букву F (ложь), за которой может следовать любой символ. Примеры записи вводимых данных в зависимости от типа элементов списка ввода приведены в табл. 1

Таблица 1

Тип данного	Внешнее представление	Внутреннее представление
Целый	15	+15
Вещественный	−12456789 12.3E12	$-0.12456789 \cdot 10^{8}$ $+0.123 \cdot 10^{14}$
Комплексный	(−45.13,13.75) (11.213D1,0.1567D2)	$-0.4513 \cdot 10^{2}+0.1375 \cdot 10^{2}$i $0.11213 \cdot 10^{3}+0.1567 \cdot 10^{2}$i
Логический	F TRUE	.FALSE. .TRUE.
Текстовый*	'ABC' 'ABCDE'	ABC␣ ABCD

* В данном примере текстовой константе соответствует элемент списка ввода вещественного типа длиной 4 байта.

Порядок следования вводимых данных задается положением соответствующих им элементов списка ввода. В качестве разделителя данных используется запятая или пробел. При вводе с перфокарточного устройства ввода конец перфокарты оказывает такое же действие,

как и разделитель. Двумя последовательными запятыми изображается отсутствующее данное (при этом значение соответствующего элемента списка ввода не изменяется). Если количество вводимых данных меньше количества элементов списка ввода, то за последним данным должна следовать дробная черта (/). В этом случае значения оставшихся элементов списка ввода не изменяются.

Пример.

```
  LOGICAL L
  READ D*8
  COMPLEX X,Y*16
5 READ(5,*) N,A,D,X,Y,L,T1
```

Если ввод данных производится с перфокарточного устройства ввода, то данные могут быть подготовлены на перфокарте в следующем виде:

```
-15,21.5,93.7D+02,(2.38,+62.7),(53D-2,2.6D3),T,'AAAB'
```

Для представления смежных и совпадающих по значению данных удобно использовать коэффициент повторения j*, где j — целая положительная константа. Например, чтобы после выполнения оператора READ(1,*) N,A,B,C переменная N получила значение 10, а переменные A, B и C — значение 21.3, данные могут быть подготовлены в виде: 10,3*21.3. С помощью конструкции j* можно указать отсутствующие данные. Например, если значения переменных A и B не должны изменяться, то данные ввода можно изобразить одним из следующих способов: 10,,,212.3 или 10,2*,212.3. Чтобы значения переменных B и C не изменились при выполнении оператора READ, данные могут быть подготовлены в виде— 15,1.3/.

При использовании оператора вывода, управляемого списком, формат данных определяется типом соответствующих им переменных в списке вывода. Данные выводятся по коду преобразования G в виде, пригодном для последующего ввода с помощью оператора ввода, управляемого списком. В качестве разделителя данных используется пробел. Комплексное данное изображается упорядоченной парой вещественных констант, разделенных запятой и заключенных в круглые скобки. Операторы вывода, управляемого списком, не позволяют вывести данные в текстовом виде (в отличие от вывода по формату с кодом преобразования A).

Глава 2

ПРАВИЛА РАБОТЫ ЗА АБОНЕНТСКИМ ПУНКТОМ

Пользователем системы разделения времени является абонент. Каждый абонент имеет в своем распоряжении абонентский пункт. АП служит для обмена информацией между ЭВМ и абонентом. АП может быть удален от ЭВМ на несколько метров или на сотни километров. В СРВ могут использоваться удаленные АП ЕС-8570, ЕС-7920 (удаленный комплекс) и устройства, непосредственно подключенные к ЭВМ через каналы ввода-вывода, например ЕС-7906, ЕС-7920 (локальный комплекс). Ввод информации с АП осуществляется с помощью алфавитно-цифровой клавиатуры, подобной клавиатуре пишущей машинки. Для АП типа ЕС-7906, ЕС-7920 вводимая и выводимая информация отображается на экране электронно-лучевой трубки. Копия изображения на экране может быть получена на печатающем устройстве. У некоторых типов АП, например ЕС-8570, отсутствует устройство отображения на электронно-лучевой трубке, поэтому вводимая и выводимая информация отображается только на печатающем устройстве. Общие принципы работы идентичны для всех типов АП. Однако имеются различия по способу ввода и редактирования данных, способу организации прерывания выполняемой работы и т. д.

2.1. ОБЩИЕ СВЕДЕНИЯ О КОМАНДАХ СРВ

Абонент получает доступ к ЭВМ с помощью языка команд СРВ, на котором он описывает свое задание. Команда СРВ является запросом на выполнение операционной системой определенных действий. Система реагирует на запросы, выполняя заданную работу и посылая сообщения на АП. СРВ обладает развитой системой команд, тем не менее пользователь Фортрана может ограничиться изучением только тех команд, которые необходимы ему для решения задач на Фортране. К числу таких команд относятся команды СРВ, позволяющие устанавливать связь с СРВ; определять характеристики сеанса работы и абонента; создавать и корректировать наборы данных с программами и данными; транслировать, редактиро-

вать, отлаживать и выполнять программы; обслуживать наборы данных; получать сведения о состоянии наборов данных; получать справочную информацию о командах СРВ; посылать сообщения другим абонентам.

Список команд СРВ, которые могут оказаться полезными при программировании на Фортране, приведен в табл. 2. Описание формата команд и их функций приводится в последующих главах и приложении 2.

Таблица 2

Назначение	Команда (сокращение)	Функция
Установка связи с СРВ	LOGON	Начинает сеанс работы
	LOGOFF	Завершает сеанс работы
Определение характеристик абонента и сеанса работы	PROFILE (PROF)	Устанавливает профиль абонента
	TERMINAL (TERM)	Определяет характеристики сеанса работы
Создание и корректировка наборов данных	EDIT (E)	Создает, изменяет и сохраняет набор данных
Обработка программы	CONVERT (CON)	Преобразовывает формат предложений Фортрана
	FORTCC	Транслирует или транслирует, редактирует и выполняет программу
	FORTSE	Транслирует программу
	RUN (R)	Транслирует, редактирует и выполняет программу
	LINK	Редактирует программу
	LOADGO (LOAD)	Редактирует и выполняет программу
	CALL	Выполняет программу
	TESTFORT (TESTF)	Выполняет отладку программы

Продолжение табл. 2

Назначение	Команда (сокращение)	Функция
Обслуживание наборов данных	ALLOCATE (ALLOC)	Распределяет набор данных
	FREE	Освобождает набор данных
	DELETE (D)	Удаляет набор данных
	RENAME (REN)	Изменяет имя набора данных
Получение сведений о состоянии наборов данных	LISTALC (LISTA)	Выдает сведения о распределенных наборах данных
	LISTCAT (LISTC)	Выдает сведения о каталогизированных наборах данных
	LISTDS (LISTD)	Выдает сведения о каталогизированных и распределенных наборах данных
Получение справочной информации о командах СРВ	HELP (H)	Выводит справочную информацию о команде СРВ
Передача сообщений	SEND (SE)	Посылает сообщение абоненту или оператору ЭВМ

Формат команд СРВ прост и не труден для изучения. Каждая команда состоит из имени, задающего запрашиваемое действие, и обычно одного или нескольких операндов. Например, команда FORTSE PROG1 SOURCE NAME(FACTOR) является запросом на выполнение трансляции с помощью транслятора Фортран SE (FORTSE — имя команды). Позиционный операнд PROG1 определяет имя набора данных с исходной программой, ключевой операнд SOURCE задает режим вывода распечаток с текстом исходной программы и диагностическими сообщениями об ошибках. Значение FACTOR в ключевом операнде NAME является именем, присваиваемым головному модулю программы.

Позиционные операнды следуют за именем команды в определенной последовательности и при описании формата команды обозначаются строчными буквами. При вводе команды такие операнды заменяются фактическими значениями. В приведенном примере позиционный операнд «имя набора данных» принимает значение PROG1 (см. 5.2.1).

Ключевые операнды записываются за позиционными в произвольном порядке. Они состоят из ключевых слов, за которыми могут следовать значения. В формате команды ключевые слова обозначаются прописными буквами. В приведенном примере для ключевого операнда NAME указано значение FACTOR. Ключевые операнды можно вводить полностью или в сокращенном виде. Допустимыми являются такие сокращения, которые обеспечивают однозначную идентификацию операндов внутри команды. Например, для ключевого операнда SOURCE минимальным является сокращение S. Допустимых сокращений значительно больше. Так, этот операнд может быть введен в виде SO, SOU, SOUR, SOURC или SOURCE.

В качестве разделителей в команде используются пробелы и запятые. Имя команды отделяется от первого операнда одним или несколькими пробелами. Значение в ключевом операнде должно следовать непосредственно за ключевым словом в скобках. Если в ключевом операнде указывается несколько значений, то они отделяются друг от друга запятыми или пробелами.

Для многофункциональных команд СРВ введены подкоманды. Каждая подкоманда является запросом на выполнение только одной функции. Синтаксис подкоманд такой же, как и синтаксис команд.

Ввод команды производится с АП. Команда набирается с любой позиции в строке и при необходимости продолжается в следующих строках. В качестве признака продолжения в конце продолжаемой строки набирается знак минус. Правила ввода строк с текстом команд специфичны для каждого типа АП. Например, для ввода строки с АП типа ЕС-7906 необходимо набрать текст и нажать клавишный переключатель ВВОД[1].

[1] Здесь и далее действия абонента за АП иллюстрируются применительно к АП типа ЕС-7906.

Ошибки, допущенные при наборе команды, могут быть исправлены сразу. Корректировка строки с текстом команды производится до передачи ее в систему. На АП типа ЕС-7906 для корректировки используется клавишный переключатель ПЕРЕВОД КУРСОРА. Если на АП отсутствует такой переключатель, его можно определить по команде СРВ PROFILE (см. 2.3).

Правила ввода подкоманд такие же, как и правила ввода команд. Значения многих операндов команд и подкоманд установлены по умолчанию (значения по умолчанию выделены при описании форматов команд). Это освобождает абонента от необходимости вводить с АП стандартные значения операндов команд и подкоманд.

Если абонент испытывает затруднения при использовании команд СРВ, он может запросить с помощью команды HELP сведения о функциях, синтаксисе и операндах нужной ему команды. Аналогичные сведения о подкомандах он может получить с помощью подкоманды HELP. Справочная информация поступает на АП и помогает абоненту быстро приобрести навыки работы в СРВ. Форматы команды и подкоманды HELP одинаковы. Поэтому описание формата дается применительно к команде HELP.

Команда HELP содержит один позиционный и четыре ключевых операнда. Позиционный операнд определяет имя команды, о которой запрашивается справочная информация. Ключевые операнды уточняют содержание справочной информации и позволяют запросить описание: основных функций команды (операнд FUNCTION), синтаксиса команды (операнд SYNTAX), отдельных операндов команды (операнд OPERANDS), всей команды в целом (операнд ALL).

Операнды команды, о которых запрашивается справочная информация, задаются с помощью соответствующих им ключевых слов в операнде OPERANDS и в списке разделяются запятыми или пробелами. Например, чтобы получить всю информацию об операндах SOURCE и PRINT команды FORTSE в команде HELP ключевой операнд следует записать в виде OPERANDS(SOURCE,PRINT). При вводе команды HELP без операндов на АП выводится список всех команд СРВ и краткое описание функций каждой из них. Определяя только имя команды, можно получить

список операндов и описание функций и синтаксиса этой команды. Например:

```
HELP FORTSE
HELP FORTSE OPERANDS(SOURCE,PRINT)
```

По первой команде HELP абонент получит всю информацию о команде FORTSE, а по второй — только сведения об операндах SOURCE и PRINT.

2.2. СРЕДСТВА СВЯЗИ АБОНЕНТА С СРВ

Связь в режиме диалога между абонентом и СРВ осуществляется с помощью языка команд СРВ и сообщений системы. Свои запросы к СРВ на выполнение определенной работы абонент передает в виде команд. Система, используя сообщения, информирует абонента о ходе выполнения запрашиваемых им действий. СРВ обладает богатым словарем для общения с абонентом. СРВ не только сообщает абоненту о допущенных им ошибках, но и во многих случаях подсказывает ему, как их исправить. Сообщения, выдаваемые на абонентский пункт, носят различный характер и поступают от многих программных компонентов.

Режимные сообщения информируют абонента об установлении режима ввода команд и подкоманд и о готовности системы принимать команды или подкоманды. В ответ на режимное сообщение READY абонент может ввести любую команду СРВ. После вывода режимного сообщения, состоящего из имени команды, абоненту разрешен ввод любой подкоманды указанной команды.

Подсказывающие сообщения указывают на ошибки, допущенные при вводе команды или подкоманды. Эти ошибки обычно связаны с неправильным написанием ключевого операнда (ключевого слова или его значения), а также с пропуском позиционных операндов, являющихся обязательными. Например, в командах FORTSE и FORTCC, используемых для вызова трансляторов, имя набора данных с исходной программой является обязательным операндом. Если абонент опустит этот операнд, то он получит сообщение с требованием ввести его. После ввода имени выполнение команды будет продолжено. Текст сообщений, требующих ответа абонента, обычно заканчивается символом минус.

Выводом подсказывающих сообщений абонент может управлять с помощью команды СРВ PROFILE. Чтобы запретить вывод таких сообщений, в команде PROFILE следует указать операнд NOPROMPT. После вывода сообщения об ошибке выполнение команды (подкоманды), в которой была допущена ошибка, будет прекращено. Потребуется повторить ввод всей команды (подкоманды) с правильными значениями операндов. Восстановить вывод подсказывающих сообщений можно, указав в команде PROFILE операнд PROMPT.

С помощью *информационных сообщений* СРВ передает абоненту сведения о состоянии системы (например, времени начала и окончания сеанса работы), уведомляет его об ошибках, допущенных при вводе команды. Обычно такие ошибки не связаны с нарушением синтаксиса команд или подкоманд, чаще всего они свидетельствуют о несоблюдении общесистемных требований. Так, информационное сообщение будет выдано, если в команде ошибочно указано имя раздела для последовательного набора данных. После информационного сообщения следует режимное или подсказывающее сообщение, определяющее последующие действия абонента.

Абоненту предоставляется возможность посылать *оповещательные сообщения* оператору ЭВМ или другим абонентам. Для этих целей служит специальная команда СРВ SEND. Позиционным операндом команды является текст сообщения, ограниченный апострофами. Идентификаторы абонентов и операторов, которым посылается сообщение, указываются соответственно в ключевом операнде USER и OPERATOR. Например, чтобы абоненту, работающему с идентификатором TERM1, послать сообщение «ОТЛАДКА ЗАКОНЧЕНА», необходимо ввести команду:

```
SEND 'ОТЛАДКА ЗАКОНЧЕНА' USER(TERM1)
```

Подсказывающие и информационные сообщения могут быть многоуровневыми. Каждый следующий уровень содержит дополнительные сведения о причине выдачи сообщения. Текст сообщений, для которых существует следующий уровень, заканчивается символом «+». Чтобы получить сообщение очередного уровня, необходимо ввести символ «?» в первой позиции строки.

Трансляторы и библиотека программ системы Фортран, работающие в СРВ, также выдают сообщения на АП. Эти сообщения носят информационный характер и указывают на синтаксические ошибки, допущенные в исходной программе, или ошибки, обнаруженные при выполнении рабочей программы.

Получив сообщение на АП, абонент легко сможет установить, какой программный компонент его выдал, если в команде PROFILE им был предусмотрен операнд MSGID. В этом случае, кроме текста, описывающего причину ошибки, выводится идентификатор сообщения, определяющий принадлежность сообщения конкретному программному компоненту. Так, идентификаторы сообщений посредника и транслятора Фортран CC содержат префикс IGK, идентификаторы сообщений транслятора Фортран SE — префикс IGI и т. д.

Ниже приведены примеры сообщений, выдаваемых на АП в СРВ. Здесь и далее текст, вводимый абонентом, представлен строчными буквами, а сообщения, выводимые СРВ, — прописными буквами.

```
logon term1 proc(fort)
IKJ56455I TERM1 LOGON IN PROGRESS AT 16:31:41 ON
    JANUARY 29, 1981
IKJ56951I NO BROADCAST MESSAGES
(Информационные сообщения о времени начала сеанса работы и
об отсутствии сохраненных для абонента сообщений)
READY
(Режимное сообщение о возможности ввода команды)
fortcc
IKJ56700A ENTER DATA SET NAME —
(Подсказывающее сообщение, запрашивающее ввод обязательного
операнда — имени набора данных с исходной программой)
bibl(a) (Ввод имени набора данных)
. . .
fortcc progm source
IKJ56712I INVALID KEYWORD, SOURCE
(Информационное сообщение о недопустимости операнда SOURCE)
IKJ56703A REENTER —
(Подсказывающее сообщение о повторном вводе операнда)
fixed (Ввод верного операнда)
. . .
READY
fortse 'bibl(c)'
IHM78106I MEMBER C NOT IN DATA SET BIBL
(Информационное сообщение об отсутствии в наборе данных с
именем BIBL раздела с именем C)
IKJ56703A REENTER —
(Подсказывающее сообщение, запрашивающее ввод правильного
имени раздела вместе с именем набора данных)
'bibl(cd)' (Ввод нового имени)
```

```
SE COMPILER ENTERED
```
(Информационное сообщение о начале работы транслятора)

```
logoff
IKJ56470I TERM1 LOGGED OFF TSO AT 16:42:53 ON
    JANUARY 29,1981+
```
(Информационное сообщение об окончании сеанса работы)

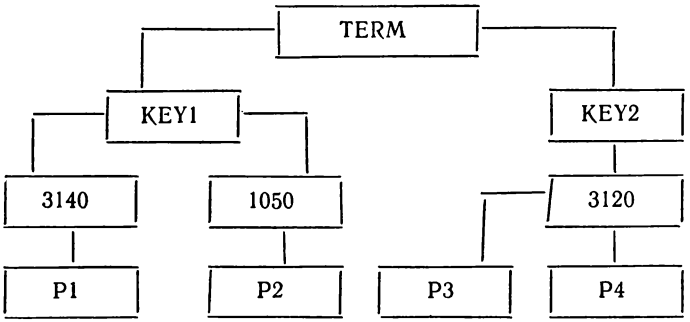
2.3. ОРГАНИЗАЦИЯ СЕАНСА РАБОТЫ

Абонентом СРВ может стать любой пользователь, который изучил правила работы за АП и получил от администрации ЭВМ разрешение на использование СРВ. Каждый абонент СРВ определяется в системе с помощью уникального имени, называемого *идентификатором.* Под этим именем абонент становится известным системе. Чтобы обеспечить возможность работы с заданным идентификатором только определенным абонентам, необходимо с этим идентификатором связать *пароли.* Абонент получит разрешение на использование идентификатора, если он правильно укажет один из паролей. Для сбора различной статистической информации о работе абонента (например, об использованном времени работы) вводятся учетные номера. Учетный номер не указывается, если в СРВ не ведется обработка учетной информации. При выполнении заданий абонента требуются определенные ресурсы — наборы данных, программы и т. д. Описание этих ресурсов производится операторами управления заданиями EXEC и DD. Эти операторы оформляются в виде процедур, аналогичных каталогизированным процедурам при работе в пакетном режиме, и записываются в системный набор данных SYS1.LOGON. Каждая такая процедура, называемая процедурой LOGON, идентифицируется уникальным именем. Требования к процедурам LOGON в зависимости от специфики решаемых на Фортране задач приводятся в гл. 8.

Идентификатор, пароли, учетные номера и имена процедур LOGON являются характеристиками каждого конкретного абонента и хранятся в наборе данных SYS1.UADS. Этот набор данных создается при генерации СРВ, но может изменяться во время функционирования системы администрацией ЭВМ. Характеристики абонента связаны в индексную структуру. На самом высоком уровне находится идентификатор абонента. С каждым

идентификатором связан один пароль или несколько паролей. Для каждого пароля указывается один учетный номер или несколько. Одна процедура или несколько процедур LOGON определяются для каждого учетного номера.

Рассмотрим следующий пример:



Абонент, идентифицируемый именем TERM, работает с двумя паролями KEY1 и KEY2. Если он использует пароль KEY1, то учетная информация может вестись по двум учетным номерам: 3140 и 1050. Если абонент выбрал учетный номер 3140, то он может запросить выполнение только процедуры LOGON с именем P1. Пароли и учетные номера являются необязательными характеристиками абонента. Их наличие определяется конкретными условиями эксплуатации СРВ. В простейшем случае характеристики абонента могут состоять только из идентификатора абонента и имени одной процедуры LOGON.

Свои характеристики абонент сообщает системе, когда он начинает работу в СРВ. Основной единицей работы в СРВ является сеанс работы, представляющий последовательность команд, введенных абонентом, и ответов системы, заключенных между командами LOGON и LOGOFF или между двумя командами LOGON. Чтобы начать сеанс работы, абонент должен включить абонентский пункт и ввести команду LOGON. В команде LOGON абонент определяет свои характеристики в виде одного позиционного и двух ключевых операндов. Позиционный операнд содержит идентификатор абонента и пароль, отделенный от идентификатора дробной чертой. Идентификатор и пароль долж-

ны соответствовать характеристикам абонента в наборе SYS1.UADS. Пароль обязателен только в том случае, когда идентификаторы абонента защищены от использования другими абонентами. Ключевые операнды содержат учетный номер (операнд ACCT) и имя процедуры LOGON (операнд PROC). Для приведенного выше примера можно использовать следующую команду LOGON:

LOGON TERM/KEY ACCT(1050) PROC(P2).

В начале сеанса работы после ввода команды LOGON абонент должен выполнить некоторые подготовительные действия для задания характеристик сеанса работы, например, определить способ вызова прерывания по сигналу ВНИМАНИЕ. Прерывание по сигналу ВНИМАНИЕ используется для установки связи абонента с СРВ, так как дает ему возможность прервать обработку и выполнить некоторые действия. Необходимость в таких действиях может возникнуть, например, в случае зацикливания программы или при появлении у абонента затруднений в правильном ответе на запросы системы. Прерывание по сигналу ВНИМАНИЕ происходит в результате нажатия специального клавишного переключателя на клавиатуре АП. Если такой переключатель отсутствует, абонент может определить его в виде некоторой последовательности не более чем из четырех символов. Эти символы указываются в операнде INPUT команды TERMINAL. Например, запись INPUT(%) означает, что ввод символа «%» должен интерпретироваться как запрос на прерывание.

Прежде чем ввести указанную последовательность символов, абонент должен получить управление на АП. Если система готова вводить данные (например, она ждет ответа абонента), то запрос на прерывание может быть выдан немедленно, иначе необходимо дождаться сообщения вида***, извещающего абонента о предоставляемой ему возможности вызвать прерывание. Сообщение*** выводится по концу экрана, а также по истечении некоторого интервала времени, который может измеряться в секундах (операнд SECONDS в команде TERMINAL) или задаваться в виде количества строк, выведенных на АП (операнд LINES). В первом случае после истечения указанного промежутка времени, если не выполнялись операции ввода-вывода

на АП, СРВ предоставит возможность вызвать прерывание по сигналу ВНИМАНИЕ. Во втором случае такая возможность будет предоставлена после того, как заданное количество строк будет выведено на АП. Способы вызова прерывания по сигналу ВНИМАНИЕ зависят от конкретного типа АП. Так, для АП типа ЕС-7906 и ЕС-7920 можно задать только временной интервал (операнд SECONDS в команде TERMINAL), для АП ЕС-8570 возможны оба способа задания (операнды SECONDS и LINES). Например, указав операнд SECONDS(60), абонент обеспечит себе возможность вызывать прерывание через каждые 60 секунд. Получив сообщение ***, абонент может вызвать прерывание вводом определенной им в команде TERMINAL последовательности символов. Если в момент прерывания выполнялась команда, не имеющая подкоманд, то выводится символ «!» и режимное сообщение READY. Если прерывается выполнение подкоманды, то выводится символ «!» и имя команды, к которой относится подкоманда. Перейти от режима ввода подкоманд в режим ввода команд можно, если повторно вызвать прерывание по сигналу ВНИМАНИЕ. После прерывания по сигналу ВНИМАНИЕ можно выполнить различные действия, например, ввести другую команду или подкоманду. Чтобы проигнорировать прерывание, достаточно ввести пустую строку.

Пример. Иллюстрируется использование прерывания по сигналу ВНИМАНИЕ на АП типа ЕС-7906.

Шаг 1. Для АП типа ЕС-7906 запрашивается возможность вызывать прерывание по сигналу ВНИМАНИЕ через каждые 60 секунд при вводе символа «%» (указываемое в операнде SECONDS значение должно быть кратно 10).

```
terminal input(%) seconds(60)
READY
```

Шаг 2. Производится трансляция и выполнение программы из набора данных с именем PROGM по команде FORTCC. При выполнении программы происходит зацикливание. Через каждые 60 секунд на АП выводится сообщение*** и управление передается абоненту для принятия им решения.

```
fortcc 'progm'
***
(пустая строка)
```

. . .

Шаг 3. Чтобы прекратить выполнение программы, абонент вызывает прерывание вводом символа «%». В ответ на запрос абонента система выводит символ «!» и сообщение READY, указывающее, что можно вводить другие команды, необходимые, например,

для корректировки программы и устранения причин ее зацикливания.

```
***
%
!
READY
```

К числу подготовительных действий, которые обычно должен выполнить абонент в начале сеанса работы, относится установка так называемого профиля абонента. Профиль абонента определяет способы исправления ошибок при вводе команд и данных, наличие идентификаторов в сообщениях, возможность вывода подсказывающих и приема оповещательных сообщений. При включении пользователя в число абонентов СРВ для него устанавливается стандартный профиль. Во время сеанса работы абонент может изменить профиль с помощью команды PROFILE. Любое изменение профиля абонента остается действующим и на время последующих сеансов работы. Все операнды команды PROFILE являются ключевыми. Ключевой операнд CHAR определяет символ удаления символа при вводе информации с АП. Например, чтобы использовать «!» в качестве символа удаления, в команде PROFILE следует указать CHAR(!). Тогда набранная на АП команда FORTCS!C будет воспринята как FORTCC.

Ключевой операнд LINE служит для определения символа удаления строки при вводе информации с АП. Например, если в команде PROFILE указать LINE(%), то набранная строка с информацией FORTCC PROGM% будет проигнорирована.

Ключевые операнды PROMPT и NOPROMPT соответственно устанавливают или отменяют режим вывода подсказывающих сообщений об ошибках, допущенных при вводе команды или подкоманды. Рекомендуется всегда работать в режиме PROMPT, так как в этом случае абонент может исправить ошибку без повторного ввода всей команды или подкоманды.

Ключевые операнды MSGID и NOMSGID соответственно задают вывод сообщений с идентификаторами и без идентификаторов. Наличие идентификатора позволяет определить компонент, который выдал соответствующее сообщение.

Ключевые операнды INTERCOM и NOINTERCOM разрешают и запрещают абоненту прием сообщений от других абонентов.

```
PROFILE CHAR(!) LINE(%) PROMPT MSGID
```

В этой команде для корректировки ошибок при вводе с АП выбраны символы «!» и «%». Ключевые операнды PROMPT и MSGID запрашивают вывод подсказывающих сообщений, а также любых других сообщений вместе с идентификаторами.

После выполнения подготовительных действий, определяемых в командах TERMINAL и PROFILE, абонент может приступить к непосредственному решению своей задачи.

Завершение сеанса работы абонента происходит по команде LOGOFF, не имеющей операндов. Если после окончания сеанса работы необходимо начать новый сеанс работы, используя, например, другое имя процедуры LOGON, то вместо команды LOGOFF достаточно сразу ввести команду LOGON. В этом случае во время нового сеанса работы будут действовать характеристики АП, установленные в предыдущем сеансе работы.

2.4. ОСОБЕННОСТИ ИСПОЛЬЗОВАНИЯ АП В КАЧЕСТВЕ УСТРОЙСТВА ВВОДА-ВЫВОДА В РАБОЧЕЙ ПРОГРАММЕ

В СРВ данные рабочей программы могут поступать не только из наборов данных на дисках, но и непосредственно с АП. Использование этой возможности оказывается полезным при отладке программы, а также при решении задач, у которых невелик объем входной и выходной информации.

Ввод и вывод информации на АП может производиться с помощью операторов форматного ввода-вывода или операторов ввода-вывода, управляемого списком. В первом случае необходимо кодировать объявление FORMAT; во втором случае вместо метки объявления FORMAT в операторе READ или WRITE указывается звездочка (*). При использовании ввода-вывода, управляемого списком, формат данных определяется типом соответствующих им элементов списка ввода-вывода.

Правила ввода данных с АП с помощью форматного ввода такие же, как и для других носителей: данные размещаются в строго определенных объявлением FORMAT позициях строки.

Пример 1.

```
      LOGICAL L
5     READ(5,1) N,A,B,L,T
1     FORMAT(I3,F5.2,E8.1,L4,A4)
```

В соответствии с указанным форматом данные должны размещаться абонентом в позициях строки АП следующим образом:

```
␣1521.151.256E+2TRUETEXT
```

Рекомендуется перед операторами ввода включать предложения, обеспечивающие вывод абоненту сообщений, уточняющих формат вводимых данных. В приведенном примере 1 перед оператором READ полезно закодировать следующие предложения:

```
      WRITE(6,2)
2     FORMAT('␣ВВЕДИТЕ ДАННЫЕ ДЛЯ N,A,B,L,T                    В
     * ФОРМАТЕ I3,F5.2,E8.1, L4,A4')
```

Получив запрос на ввод по оператору READ, абонент набирает данные на клавиатуре АП в соответствии с требованиями объявления FORMAT.

При выводе данных на АП независимо от значения управляющего символа, указанного в объявлении FORMAT, строки размещаются через один интервал.

Правила ввода с АП с помощью операторов ввода, управляемого списком, освобождают абонента от необходимости размещать данные в фиксированных позициях строки. Вводимые данные записываются в одной или нескольких строках и разделяются запятыми или пробелами.

Пример 2.

```
      LOGICAL L
5     READ(5,*)N,A,B,L,T
```

Данные вводятся в одной строке:

```
15,21.15,1.256E+2,TRUE,'TEXT'
```

Те же данные могут вводиться в нескольких строках, например, в виде:

```
15,21.15,1.256E+2
TRUE,'TEXT'
```

Во время выполнения оператора ввода на АП выдается подсказывающее сообщение, содержащее символ «?» и метку оператора ввода, если он помечен. В примере 2 сообщение будет иметь вид: ? 00005. Запрос на ввод выдается до тех пор, пока не будут заполнены все элементы списка ввода, или пока во вводимых данных не встретится символ «/».

Раздел II

СОЗДАНИЕ И ОБСЛУЖИВАНИЕ НАБОРОВ ДАННЫХ

Глава 3

СОЗДАНИЕ И КОРРЕКТИРОВКА НАБОРОВ ДАННЫХ

В режиме пакетной обработки данные обычно готовятся на перфокартах. Для последующего хранения и сопровождения данные переносятся с перфокарт в наборы данных на дисках или магнитных лентах. В режиме разделения времени наборы данных можно создавать на дисках непосредственно с АП. Абонент вводит данные, набирая их на клавиатуре АП, и записывает в набор данных, размещенный на дисках. В этом случае отпадает необходимость в предварительной подготовке данных на перфокартах. Кроме того, использование АП предоставляет абоненту и ряд других преимуществ, связанных с возможностью оперативно корректировать и обновлять данные в наборах данных. В системе программирования Фортран можно создавать наборы данных, содержащие как исходные тексты программ, так и входные данные программ. Наборы данных, обрабатываемые в СРВ, имеют только последовательную или библиотечную организацию.

3.1. ИМЕНОВАНИЕ НАБОРОВ ДАННЫХ

В операционной системе ОС ЕС наборы данных идентифицируются простыми и составными именами. Простое имя содержит не более 8 буквенно-цифровых символов (A — Z, 0—9), первым символом должна быть буква. Составное имя образуется из нескольких простых имен, разделенных точкой. Например, имя набора данных SOURCE является простым, имя SOURCE.FORT — составным, состоящим из двух простых имен SOURCE и FORT. Длина составного имени,

включая точки, не должна превышать 44 символа. Если набор данных имеет библиотечную организацию, то каждый раздел идентифицируется простым именем длиной не более 8 символов. Имя раздела заключается в круглые скобки и записывается после имени набора данных. Например, имя PROGM.FORT(PART1) определяет раздел PART1 в наборе данных PROGM.FORT. В системе разделения времени имена наборов строятся по этим же правилам, но дополняются специальными уточнителями.

Идентифицирующий уточнитель является идентификатором абонента, которому принадлежит набор данных, и записывается в начале имени.

Описательный уточнитель определяет вид данных в наборе данных и записывается в конце имени набора данных. В СРВ описательные уточнители зафиксированы. Например, для наборов данных с исходными программами на языке Фортран используется описательный уточнитель FORT, для наборов данных рабочей программы — уточнитель DATA. Имя со всеми уточнителями называется полным уточненным именем. Соглашения об именовании наборов данных упрощают задание имен наборов данных в командах языка СРВ. Например, к набору данных TERM.PROGM.FORT, содержащему исходную программу на Фортране, абонент, работающий с идентификатором TERM, может обратиться, указав только имя PROGM.

В качестве идентифицирующего уточнителя используется идентификатор абонента, заданный в команде LOGON, а описательный уточнитель определяется на основании информации о виде данных, указанной в команде СРВ. Описательный уточнитель формируется не всеми командами. Например, при работе с командой ALLOCATE абонент должен указать его сам.

В табл. 3 приводится список команд и описательных уточнителей, присваиваемых по умолчанию в СРВ именам наборов данных при работе с указанными командами.

В тех случаях, когда абонент указывает полное уточненное имя набора данных или имя, построенное не по правилам СРВ (например, SYS1.FORTLIB), при вводе команды он должен ограничивать его апострофами. Такой способ задания имени обычно использу-

Таблица 3

Команда СРВ	Описательный уточнитель	Содержимое набора данных
FORTCC FORTSE RUN EDIT CONVERT TESTFORT	FORT	Исходные программы на Фортране
FORTCC FORTSE LINK LOADGO TESTFORT	OBJ	Объектные модули программы
LINK CALL TESTFORT	LOAD	Загрузочные модули программы
FORTSE TESTFORT	LIST	Распечатки результатов трансляции Распечатка отладочной информации
LINK	LINKLIST	Распечатки результатов редактирования
LOADGO	LOADLIST	Распечатки результатов редактирования

ется для доступа к наборам данных, созданным другими абонентами или вне СРВ.

Примечание. При описании форматов команд языка СРВ операнд «имя набора данных» обычно используется для идентификации как всего набора данных, так и отдельных разделов библиотечного набора данных.

Пример. Исходная программа находится в наборе данных с именем TERM1.SOURCE.FORT. Чтобы выполнить трансляцию этой программы, абонент, работающий с идентификатором TERM1, может ввести команду FORTSE SOURCE или FORTSE 'TERM1. SOURCE.FORT'.

Абонент, работающий с идентификатором TERM2, это же действие может выполнить только по команде FORTSE 'TERM1.SOURCE.FORT', указав в ней полное имя набора данных.

3.2. СОЗДАНИЕ НАБОРОВ ДАННЫХ

Основным средством создания и корректировки наборов данных в СРВ является команда EDIT. Эта команда позволяет обслуживать наборы данных с последовательной и библиотечной организациями. Записи в наборах данных могут быть фиксированной или переменной длины, причем размер записи не должен превышать 255 байт, а размер блока — 1680 байт.

3.2.1. Формат команды EDIT

Команда EDIT имеет следующий формат:

$$\left\{\begin{matrix}\text{EDIT}\\ \text{E}\end{matrix}\right\}\text{имя набора данных}\ \left[\begin{matrix}\text{NEW}\\ \underline{\text{OLD}}\end{matrix}\right]\left[\begin{matrix}\text{FORTSE}\\ \text{FORTCC}\\ \text{DATA}\end{matrix}\left[\left(\left\{\begin{matrix}\text{FIXED}\\ \underline{\text{FREE}}\end{matrix}\right\}\right)\right]\right]$$

$$\left[\begin{matrix}\text{SCAN}\\ \underline{\text{NOSCAN}}\end{matrix}\right]\left[\begin{matrix}\text{NUM}\\ \underline{\text{NONUM}}\end{matrix}\right]\ [\text{BLOCK(число)}]\,[\text{LINE(число)}]$$

В поле операндов команды EDIT можно записать один позиционный операнд и до шести ключевых операндов.

Позиционный операнд задает имя набора данных и является обязательным. *Имя набора данных* определяет имя последовательного набора данных или имя библиотечного набора данных вместе с именем раздела, в который производится запись информации. Как видно из примеров, приведенных в табл. 4, правила именования наборов данных в СРВ упрощают задание их имен в команде EDIT.

Ключевые операнды содержат сведения о наборе данных.

Операнды NEW и OLD определяют состояние набора данных перед выполнением команды EDIT. Операнд NEW указывает, что создается новый последовательный набор данных или новый раздел в существующем или новом библиотечном наборе данных. Операнд OLD задается для существующего набора данных или раздела набора данных и служит указанием на то, что команда EDIT используется для корректировки дан-

Таблица 4

Имя набора данных	Идентификатор абонента	Имя раздела	Организация*	Задание имени в команде EDIT
TERM.SOURCE.FORT	TERM	—	PS	SOURCE
TERM.INPUT.DATA	TERM	—	PS	INPUT
TERM.BIBLIO.FORT	TERM	PART1	PO	BIBLIO(PART1)
TERM.BIBLIO.FORT	TERM	TEMPNAME	PO	BIBLIO
TERM.SOURCE.FORT	TERM1	—	PS	'TERM.SOURCE.FORT'
TERM.BIBLIO.FORT	TERM1	PART1	PO	'TERM.BIBLIO.FORT(PART1)'
TERM.INPUT.DATA	TERM1	TEXT	PO	'TERM.INPUT.DATA(TEXT)'
TERM.OUTPUT.DATA	TERM1	—	PS	'TERM.OUTPUT.DATA'

* PS — последовательная организация набора данных, PO — библиотечная организация набора данных.

ных. Тип набора данных определяется одним из операндов FORTSE, FORTCC и DATA. Операнд FORTSE указывает, что набор данных содержит исходную программу на Фортране, которую предполагается в дальнейшем использовать как входную информацию для транслятора Фортран SE. Операнд FORTCC(FIXED) или FORTCC(FREE) задается для набора данных с исходной программой на Фортране, обработку которой планируется выполнять транслятором Фортран CC. Значение FIXED указывает, что предложения Фортрана вводятся в стандартном, а значение FREE — в свободном формате. Более детально допустимые форматы ввода предложений Фортрана описаны в 3.2.3. Операнд DATA задается при создании или корректировке набора данных, содержащего входную информацию для рабочей программы.

В том случае, когда в набор данных записывается программа на Фортране, имеется возможность выполнить частичную проверку ее синтаксиса непосредствен-

но на этапе создания. Установка этого режима производится с помощью операнда SCAN. Каждое предложение исходной программы проверяется на наличие синтаксических ошибок. Если в предложении обнаружены ошибки, выдаются диагностические сообщения, после чего абоненту предоставляется возможность повторить ввод предложения (см. 3.4). Если создается набор данных для рабочей программы, то в команде EDIT не следует указывать операнд SCAN. По умолчанию предполагается NOSCAN.

При создании набора данных можно запросить режим нумерации строк набора данных, задав операнд NUM. Строка является единицей информации, набранной на клавиатуре АП и записанной в набор данных нажатием клавишного переключателя ВВОД. Нумерация выполняется с номера 10 для первой строки с шагом 10 для всех последующих строк. В наборах данных типа FORTSE, FORTCC(FIXED) и DATA номер строки записывается в последних 8 позициях, а типа FORTCC(FREE) — в первых 8 позициях записи. Операнд NUM рекомендуется всегда использовать при создании наборов данных с исходными программами, так как он обеспечивает определенные удобства при корректировке программы в процессе ее отладки. При указании операнда NONUM строки набора данных не нумеруются, поэтому его рекомендуется указывать для наборов данных с входными данными рабочей программы, так как, во-первых, такие наборы данных реже подвергаются корректировке, а, во-вторых, при написании предложений FORMAT, используемых для ввода данных, не потребуется учитывать наличие номеров строк в записях.

Операнды BLOCK и LINE позволяют для нового набора данных указать максимальные размеры блоков и записей (строк) в байтах. Характеристики существующего набора данных изменить нельзя. Значения операндов BLOCK и LINE в зависимости от типа набора данных приведены в табл. 5.

Примеры.

```
EDIT SOURCE(PART) NEW FORTSE SCAN
EDIT TEXT NEW DATA BLOCK(400)  LINE(100)
```

Первая команда EDIT запрашивает создание нового раздела PART в существующем или новом (опе-

Таблица 5

Тип набора данных	Операнд LINE (в байтах)		Операнд BLOCK (в байтах)		Формат записей
	по умолчанию	определяемый	по умолчанию	определяемый	
DATA	80	⩽255	1680	⩽1680	Фиксированный
FORTSE	80	80	400	⩽ 400	Фиксированный
FORTCC(FIXED)	80	80	400	⩽ 400	Фиксированный
FORTCC(FREE)	255	Не задается	1680	⩽1680	Переменный

ранд NEW) библиотечном наборе данных с именем TERM.SOURCE.FORT (предполагается, что идентификатор абонента — TERM). Раздел планируется для записи в него исходной программы для последующей обработки ее транслятором Фортран SE (операнд FORTSE). При вводе должна быть выполнена синтаксическая проверка предложений (операнд SCAN). Набор данных компонуется из записей и блоков со стандартными размерами соответственно 80 и 400 байт, установленными по умолчанию.

Вторая команда EDIT используется для создания нового (операнд NEW) последовательного набора данных с именем TERM.TEXT.DATA, содержащего данные рабочей программы (операнд DATA). Размеры записей и блоков задаются в операндах LINE и BLOCK равными соответственно 100 и 400 байтам.

3.2.2. Режимы работы команды EDIT

Команда EDIT работает в двух режимах: режиме ввода и режиме редактирования. В режиме ввода создается новый набор данных или новый раздел набора данных. В режиме редактирования выполняется корректировка существующего набора данных или раздела с помощью подкоманд команды EDIT.

Режим ввода устанавливается, если абонент указал в команде EDIT имя нового набора данных или раздела и операнд NEW.

После установления режима ввода абоненту выдается сообщение INPUT, запрашивающее ввод данных. В следующем примере операнд NEW в команде EDIT

определяет, что абонент, работающий с идентификатором PETROV, создает набор данных с именем PETROV.PROG.FORT, содержащий исходную программу. Последующую обработку программы он планирует выполнять транслятором Фортран SE.

```
edit prog new fortse
INPUT
```

Режим редактирования устанавливается, если абонент указал в команде EDIT имя существующего набора данных или раздела и операнд OLD. После установления режима редактирования абоненту выдается сообщение EDIT, позволяющее вводить подкоманды для корректировки набора данных. В следующем примере операнд OLD в команде EDIT указывает, что абонент планирует корректировать данные в созданном им ранее наборе данных с именем SOURCE. Набор данных содержит исходную программу, трансляция которой будет выполняться транслятором Фортран CC.

```
edit 'source' old fortcc(free)
EDIT
```

Переход из одного режима работы команды EDIT к другому производится автоматически при наступлении некоторых условий или по запросу абонента. Команда EDIT автоматически переходит из режима ввода в режим редактирования, если нет места в наборе данных для ввода новых записей или обнаружена ошибка при синтаксической проверке предложения. Абонент может изменить режим ввода на режим редактирования, если введет пустую строку или вызовет прерывание по сигналу ВНИМАНИЕ. Переход из режима редактирования в режим ввода возможен только в случае ввода абонентом пустой строки или подкоманды INPUT команды EDIT.

3.2.3. Запись исходной программы в набор данных

Запись исходной программы в набор данных выполняется в режиме ввода команды EDIT. Команда EDIT допускает два формата представления предложений при вводе их с АП: стандартный и свободный. Ввод программы в стандартном формате производится при задании операнда FORTSE или FORTCC(FIXED), а ввод в свободном формате — при задании операнда FORTCC или FORTCC(FREE).

Для транслятора Фортран SE предложения должны быть подготовлены только в стандартном формате. Транслятор Фортран CC допускает оба формата.

Ввод программы в стандартном формате. При вводе исходной программы в стандартном формате каждое предложение размещается в строке АП в том виде, в каком оно записывалось бы на бланке кодирования (метка — позиции 1—5 строки, текст предложения — позиции 7—72). Чтобы продолжить предложение в следующей строке, необходимо в конце продолжаемой строки набрать знак минус. При вводе строки продолжения в позиции 6 следует набрать символ, отличный от пробела или нуля, а затем текст предложения. Для одного предложения Фортрана допускается не более 19 строк продолжения. При записи предложения в набор данных минусы в конце строк продолжения удаляются командой EDIT. По операнду NUM вводимым строкам присваиваются порядковые номера, которые записываются в позициях 73—80 строки. Каждая строка в наборе данных представляется в виде записи длиной 80 байт. Записи могут объединяться в блоки. Длина блока задается в операнде BLOCK и не должна превышать 400 байт.

Пример. Иллюстрируется запись в набор данных программы, представленной в стандартном формате:

```
C ПОДПРОГРАММА
      FUNCTION S(X,Y)
      IF(Y)1,2,1
    1 S=X**3+X*4-3.14*X*
     +Y+2*X/Y
      RETURN
    2 S=0.
      RETURN
      END
```

Шаг 1. Ввод команды EDIT. После получения сообщения READY о возможности вводить команды для создания нового набора данных PROGM с исходной программой вводится команда EDIT. Предполагается, что трансляция программы будет выполняться транслятором Фортран SE. Запрашивается синтаксическая проверка предложений при вводе.

```
READY
edit 'progm' new fortse scan
INPUT
00010
```

Сообщение INPUT выдается системой и информирует абонента о том, что можно начинать ввод программы. Первой строке присваивается порядковый номер 00010 (в команде EDIT по умолчанию предполагается режим NUM).

Шаг 2. Набор первой строки, содержащей комментарий, и ввод ее нажатием клавишного переключателя ВВОД.

```
00010 с подпрограмма
00020
```

Так как действует режим NUM, команда EDIT после ввода строки выдает номер следующей строки. В данном случае следующая строка получает номер 00020.

Шаг 3. Ввод остальных операторов программы.

```
00020        function s(x,y)
00030        if(y)1,2,1
00040  1     s=x**3+x*4-3.14*x*-
00050       +y+2*x/y
00060        return
00070  2     s=0.
00080        return
00090        end
00100
```

Оператор присваивания записывается в двух строках, поэтому в конце строки с номером 00040 набирается знак минус для указания строки продолжения. В строке с номером 00050 в позиции 6 указывается символ строки продолжения. При записи этого оператора в набор данных команда EDIT удаляет знак минус в конце строки 00040.

Шаг 4. Установка режима редактирования. После ввода всей программы следует ввести пустую строку (нажать клавишный переключатель ВВОД, не набирая никакой информации). Устанавливается режим редактирования. В этом режиме можно вводить любые подкоманды команды EDIT.

```
00100 (пустая строка)
EDIT
```

Шаг 5. Сохранение набора данных. Чтобы сохранить набор данных для использования другими командами СРВ, следует ввести подкоманду SAVE, имеющую формат:

$$\left\{\begin{matrix}\text{SAVE}\\ \text{S}\end{matrix}\right\}\ [\text{имя набора данных}]$$

По подкоманде SAVE сведения о наборе данных сохраняются в каталоге операционной системы.

Если в подкоманде SAVE операнд опущен, то набор данных сохраняется под тем именем, которое ему было присвоено в команде EDIT. Обычно подкоманда SAVE без операнда используется для сохранения нового набора данных. В нашем примере набор данных сохраняется под именем PROGM.

```
save
SAVED
```

Сообщение SAVED, которое выдается системой, указывает, что запрос абонента на сохранение набора данных удовлетворен.

Шаг 6. Переход в режим ввода команд СРВ. Чтобы завершить выполнение команды EDIT, необходимо ввести подкоманду END. По подкоманде END действие команды EDIT заканчивается, и система переходит из режима ввода подкоманд EDIT в режим ввода команд. О возможности вводить новые команды система уведомляет абонента сообщением READY.

```
end
READY
```

Ввод программы в свободном формате. Представление предложений в свободном формате более удобно для ввода программы с АП, так как не требует соблюдения жестких правил записи предложения в позициях строки. Предложение может начинаться с любой позиции строки и содержать в начале метку. От остальной части строки метку отделять пробелом необязательно. Предложение можно продолжить в следующих строках. Для этого в конце каждой продолжаемой строки набирается знак минус. При записи в набор данных минусы в конце не удаляются. В случае продолжения литерала его необходимо набирать, начиная с первой позиции строки продолжения. Для одного предложения Фортрана допускается не более 19 строк продолжения, причем общая длина предложения не должна превышать 1326 символов.

Признаком строки комментариев служит символ кавычки (″) в первой позиции строки. Комментарии нельзя размещать между строками продолжения предложения. Все комментарии, относящиеся к одному предложению, должны следовать за последней строкой предложения. Каждая строка представляется в наборе данных в виде записи переменной длины размером не более 255 байт. Длина блока задается в операнде BLOCK и не должна превышать 1680 байт.

Пример. Иллюстрируется запись в набор данных программы в свободном формате. Абонент работает с идентификатором USE.

```
″ПОДПРОГРАММА
FUNCTION S(X,Y)
IF(Y)1,2,1
1S=X**3+X*4—3.14*X* —
Y+2*X/Y
RETURN
2S=0.
RETURN
END
```

Шаг 1. Ввод команды EDIT для создания нового набора данных USE.SOURCE.FORT с исходной программой. Трансляция программы планируется транслятором Фортран CC. Для этого указывается операнд FORTCC(FREE). При вводе программы должна быть выполнена синтаксическая проверка предложений Фортрана.

```
READY
edit source new fortcc(free) scan
INPUT
00010
```

Шаг 2. Ввод первой строки с комментарием.

```
00010 "подпрограмма
00020
```

Шаг 3. Ввод остальных предложений программы.

```
00020 function s(x,y)
00030 if(y)1,2,1
00040 1s=x**3+ x*4—3.14*x*—
00050 y+2*x/y
00060 return
00070 2s=0.
00080 return
00090 end
00100
```

Шаг 4. Установка режима редактирования.

```
00100 (пустая строка)
EDIT
```

Шаг 5. Сохранение набора данных.

```
save
SAVED
```

Шаг 6. Переход в режим ввода команд СРВ.

```
end
READY
```

3.2.4. Преобразование формата предложений

Как уже отмечалось, транслятор Фортран CC может обрабатывать исходные программы, подготовленные в стандартном или в свободном формате. Транслятор Фортран SE воспринимает предложения только в стандартном формате. В некоторых случаях может возникнуть необходимость изменения формата предложений. Например, программа, подготовленная в свободном формате и отлаженная с помощью транслятора Фортран CC, должна быть обработана транслятором Фортран SE для получения эффективного объектного модуля и документирования результатов трансляции. Для этого следует преобразовать исходную программу из

свободного формата в стандартный. Или наоборот, в программу, подготовленную в стандартном формате, планируется вносить большое количество изменений, которые удобнее вводить в свободном формате. В этом случае также потребуется выполнить преобразование формата предложений, но уже из стандартного в свободный. Изменить формат предложений Фортрана можно по команде CONVERT.

Команда CONVERT имеет формат:

```
{CONVERT}                                          {FREE }
{CON    } имя набора данных OUT(имя набора данных) [FIXED] FORTCC
```

Имя набора данных с исходной программой указывается с помощью позиционного операнда. Имя набора данных с преобразованной программой задается ключевым операндом OUT. Это имя идентифицирует новый последовательный набор данных или новый раздел библиотечного набора данных.

Ключевой операнд FORTCC служит для определения типа набора данных, содержащего предложения языка Фортран. Ключевые операнды FREE и FIXED задают способ преобразования формата предложений: FREE означает, что должно быть выполнено преобразование предложений из стандартного формата в свободный, а FIXED — из свободного формата в стандартный.

Пример 1. Необходимо предложения в свободном формате, содержащиеся в наборе данных с именем IVANOV.PROG1.FORT и принадлежащие абоненту с идентификатором IVANOV, преобразовать к стандартному формату и записать в раздел PROG1 библиотечного набора данных с именем IVANOV.BIBLIO.FORT. Это можно выполнить командой

```
CONVERT PROG1 OUT(BIBLIO(PROG1)) FORTCC
```

Пример 2. Необходимо предложения, которые находятся в разделе SUB1 набора данных с именем SOURCE, преобразовать из стандартного формата в свободный и сохранить их в разделе PART1 набора данных с именем PROG2. Это можно выполнить командой

```
CONVERT 'SOURCE(SUB1)' OUT('PROG2(PART1)') FORTCC FREE
```

3.2.5. Создание наборов данных для рабочих программ

В СРВ с помощью команды EDIT можно создавать с АП наборы данных, содержащие входную информацию для рабочих программ. Для таких наборов данных в команде EDIT указывается тип DATA. Размер записей и блоков задается соответственно операндами LINE и BLOCK. Значение операнда LINE не должно превышать 255 байт, а операнда BLOCK — 1680 байт. По умолчанию их значения установлены равными соответственно 80 и 1680 байтам.

Набор данных компонуется из записей фиксированной длины. Если запись короче длины, указанной в операнде LINE, то она дополняется пробелами, если длиннее — то усекается справа. Наборы данных рабочих программ с записями переменной длины не могут быть созданы средствами команды EDIT. Такие наборы данных готовятся в режиме пакетной обработки. При указании параметра NUM строки с данными нумеруются. Номер строки помещается в последние 8 позиций каждой записи (это необходимо учитывать при программировании операций ввода).

По команде EDIT данные записываются в набор данных в коде ДКОИ, поэтому их следует читать с помощью операторов форматного ввода или операторов ввода, управляемого списком. Хотя команда EDIT позволяет записывать данные как в последовательные, так и в библиотечные наборы данных, использование библиотечных наборов данных при выполнении рабочей программы в СРВ невозможно.

Пример. Иллюстрируются возможности команды EDIT для записи данных в набор данных с именем ARRAY.

Шаг 1. Ввод команды EDIT.

```
edit 'array' new data line(24) block(120) nonum
INPUT
```

В команде EDIT указываются основные сведения о наборе данных: имя набора данных (ARRAY), тип набора данных (DATA), состояние набора данных, определяющее его как новый (NEW), размер записи — 24 байта и размер блока — 120 байт.

Шаг 2. Построчный ввод данных. Так как в команде EDIT указан параметр NONUM, строки с данными не нумеруются. Каждая строка в наборе данных представляется в виде одной записи длиной 24 байта. Предполагается, что в программе на Фортране

данные будут читаться по формату 4F6.2. С учетом такого формата записей в каждой строке размещаются значения для четырех данных. На представление каждого данного отводится 6 позиций. Например, вводимая строка может содержать данные:

512.31␣␣1.15153.12123456

Шаг 3. Сохранение набора данных и переход в режим ввода команд.

```
save
SAVED
end
READY
```

В результате указанных действий создается последовательный набор данных с именем ARRAY. Считывание данных из этого набора данных можно выполнить, например, следующими предложениями Фортрана:

```
  DIMENSION ARRAY (5,4)
  READ(5,1) ((ARRAY(I,J),J=1,4),I=1,5)
1 FORMAT (4F6.2)
```

3.3. КОРРЕКТИРОВКА НАБОРОВ ДАННЫХ

Корректировка наборов данных в СРВ, так же как и их создание, производится в режиме редактирования с помощью подкоманд команды EDIT. Команда EDIT позволяет выполнять удаление, замену и добавление строк в набор данных, изменение последовательности символов в строках, перенумерацию строк и присваивание номеров строкам непронумерованного набора данных. Ниже приведен список подкоманд, наиболее употребительных при корректировке наборов данных.

Подкоманда (сокращение)	Функция
CHANGE (C)	Изменяет последовательность символов в строке
DELETE (D)	Удаляет строки из набора данных
END	Завершает выполнение команды EDIT
INPUT (I)	Устанавливает режим ввода
LIST (L)	Выводит строки набора данных на АП
RENUM (REN)	Перенумеровывает строки набора данных
SCAN (SC)	Выполняет синтаксическую проверку предложений Фортрана при вводе
SAVE (S)	Сохраняет набор данных после создания или корректировки

3.3.1. Построчная корректировка наборов данных

При построчной корректировке набора данных можно добавлять, заменять или удалять только целые

строки. Для такой корректировки набора данных требуется, чтобы его строки были пронумерованы. Нумерацией строк можно управлять как при создании набора данных, так и при его корректировке. При создании нового набора данных нумерация строк запрашивается с помощью операнда NUM в команде EDIT. Присвоение или изменение номеров строк уже существующего набора данных производится по подкоманде RENUM. Для наборов данных типа FORTSE, FORTCC(FIXED) и DATA номер строки заносится в последние восемь позиций записи, а для типа FORTCC(FREE) — в первые восемь позиций.

Для перенумерации всего набора данных в подкоманде RENUM следует указать два позиционных операнда: номер первой строки, назначенный набору данных, и шаг, на который должен увеличиваться номер каждой следующей строки. Если один или оба операнда опущены, их значения принимаются равными 10. Например, по подкоманде RENUM первой строке набора данных присваивается номер 10, а каждой последующей строке на 10 больше.

Подкомандой RENUM можно изменить номера строк некоторой части набора данных, начиная с указанной строки. Для этого необходимо указать новый номер строки, шаг для последующих строк и номер строки, начиная с которой производится перенумерация. Например, после выполнения подкоманды RENUM 200 5 150 строке с номером 150 будет присвоен номер 200, а каждой последующей строке — номер на 5 больше: 205, 210 и т. д.

После того как строки набора данных перенумерованы, можно выполнять корректирующие действия, используя подкоманды DELETE и INPUT. С помощью DELETE производится удаление из набора данных одной или нескольких строк. Для удаления одной строки следует указать ее номер, для удаления группы строк следует указать номер первой и последней строки из этой группы.

Пример 1. Из набора данных с именем PROG необходимо удалить строку с номером 80 и строки с номерами 200—250. Это можно выполнить с помощью последовательности действий:

```
edit 'prog' old fortse
EDIT
```

```
delete 80
delete 200 250
save
SAVED
end
READY
```

Для корректировки набора данных в команде EDIT должен задаваться операнд OLD (поскольку набор данных уже существует). Операнд FORTSE определяет тип набора данных как совокупность предложений Фортрана, подготовленных в стандартном формате. Чтобы удалить одну строку из набора данных, достаточно набрать только номер строки без имени подкоманды, в данном случае 80. Измененный набор данных можно сохранить по подкоманде SAVE. Этой подкомандой абонент всегда должен завершать последовательность корректирующих действий. Если команду SAVE опустить, изменения не попадут в исходный набор данных, а сохранятся только во временном наборе данных, к которому абонент не имеет доступа после завершения команды EDIT. Задав в подкоманде SAVE имя, можно сохранить два варианта набора данных: до корректировки — под старым именем, указанным в команде EDIT, и после корректировки — под новым именем, указанным в подкоманде SAVE. Такой режим работы удобен для сопровождения различных вариантов одной и той же программы.

Добавление или замена строк в наборе данных выполняется по подкоманде INPUT. Подкоманда INPUT имеет следующий формат:

$$\left\{ \begin{matrix} \text{INPUT} \\ \text{I} \end{matrix} \right\} \text{[номер строки[шаг]]} \left[\frac{\text{I}}{\text{R}} \right]$$

Первый операнд указывает номер строки, которая добавляется или заменяется в наборе данных. При отсутствии этого операнда строка добавляется в конец набора данных. Второй операнд указывает значение, на которое должен увеличиваться номер каждой следующей вводимой строки. По умолчанию шаг нумерации принимается равным 10. Операнды I и R определяют корректирующие действия: I означает добавление строк в набор данных, а R — их замену.

Пример 2. В набор данных с именем SOURCE1, содержащий программу в свободном формате, необходимо между строками с номерами 10 и 20 добавить две

строки (операторы WRITE(6,10)A,B и 10FORMAT (2F5.1)), присвоив им номера 12 и 15, а операторы в строках с номерами 100 и 110 заменить на A=B+C и 2FORMAT(3F5.1).

Корректировка набора данных выполняется следующим образом:

```
edit 'source 1' old fortcc(free)
EDIT
input 12 3 i
INPUT
00012 write(6,10)a,b
00015 10format(2f5.1)
00018 (пустая строка)
EDIT
input 100 r
INPUT
00100 a=b+c
00110 2format(3f5.1)
00120 (пустая строка)
EDIT
save 'source2'
SAVED
end
READY
```

В команде EDIT определяется существующий набор данных SOURCE1, содержащий программу на Фортране, подготовленную в свободном формате. Подкоманда INPUT используется дважды: первый раз для добавления двух операторов с номерами 12 и 15, второй раз — для замены двух операторов с номерами 100 и 110. Скорректированная программа сохраняется в наборе данных с именем SOURCE2. Исходный вариант этой программы продолжает существовать в наборе данных с именем SOURCE1. Сообщение INPUT информирует абонента о возможности вводить данные. Ввод программы производится в свободном формате.

Добавление и замену строк в наборе данных можно выполнять не только при работе в режиме ввода команды EDIT, но и в режиме редактирования. Каждый из этих способов имеет свои преимущества и недостатки. Режим редактирования для корректировки предпочтительнее в тех случаях, когда в набор данных добавляется или в нем заменяется одна строка. Чтобы выполнить требуемое действие, достаточно набрать номер строки и ее текст. Однако если корректируется несколько строк, то этот способ менее удобен, так как при его использовании необходимо для каждой строки

набирать ее порядковый номер. При работе в режиме ввода номер каждой строки выводится системой автоматически, и абонент должен дополнить строку только текстом. Недостатком способа корректировки в режиме редактирования является также то, что действие режима проверки синтаксиса предложений Фортрана, установленного в команде EDIT по операнду SCAN, не распространяется на корректируемые строки. Для проверки строк, введенных в режиме редактирования, следует использовать подкоманду SCAN. Более подробно об этом см. в 3.4.

В процессе создания и корректировки набора данных подкоманда LIST позволяет абоненту следить за содержимым набора данных. Используя эту подкоманду, можно вывести на АП весь набор данных или только отдельные строки. В первом случае достаточно ввести имя подкоманды без операндов. Во втором случае необходимо указать границы участка в виде номеров первой и последней строки. Например, чтобы вывести на АП строки с номерами 50—200, необходимо ввести подкоманду LIST 50 200.

Пример 3. Показываются возможности отдельных подкоманд команды EDIT для создания и корректировки набора данных. Абонент работает с идентификатором TERM.

Шаг 1. Ввод команды EDIT для записи программы на Фортране в стандартном формате в набор данных с именем TERM.MAIN.FORT.

```
edit main new fortse
INPUT
00010
```

Шаг 2. Ввод предложений программы.

```
00010    dimension a(10),b(20)
          .  .  .
00320    end
00330    (пустая строка)
EDIT
```

Шаг 3. Добавление в набор данных двух строк с номерами 12 и 17.

```
input    12 5
00012    common i(20)
00017    real*8 c j
00022    (пустая строка)
EDIT
```

Шаг 4. Перенумерация строк набора данных. Первой строке присваивается номер 10, всем последующим — номера с шагом 10.

```
renum
```

Шаг 5. Вывод на АП всего набора данных для визуальной проверки программы.

```
list
00010     DIMENSION A(10),B(20)
00020     COMMON L(20)
00030     REAL*8 C
. . .
00340     END
END OF DATA
```

Шаг 6. Сохранение программы в наборе данных с именем TERM.MAIN.FORT и переход в режим ввода команд.

```
save
SAVED
end
READY
```

3.3.2. Изменение последовательности символов в строках

В СРВ имеется возможность корректировать в наборе данных не только целые строки, но и отдельные части строк. Как и в случае построчной корректировки, набор данных должен быть пронумерован. Для изменения последовательности символов в строке предназначена подкоманда CHANGE. Подкоманда CHANGE позволяет обновить информацию в одной или нескольких строках. Ее целесообразно использовать в тех случаях, когда одни и те же изменения необходимо внести в группу строк набора данных, например, заменить в программе символическое имя или метку перехода. Для этого в команде CHANGE следует указать номер строки или границы диапазона строк, в которые вносятся изменения. Корректирующая информация задается непосредственно в подкоманде или может поступать с АП. Неизменяемая часть строки в команде CHANGE не указывается. Это позволяет избежать лишних ошибок при вводе данных с АП.

Пример 1. В строках с номерами из диапазона 60—200 символическое имя MASSIV необходимо заменить новым именем ARRAY. Указанное действие можно выполнить с помощью следующей подкоманды:

```
CHANGE 60 200 !MASSIV!ARRAY!ALL
```

Символ «!» является специальным разделителем, который должен предшествовать старой и новой последовательностям символов. Таким разделителем может быть любой символ, кроме цифры, апострофа, пробела, запятой, точки с запятой, круглых и квадратных скобок, звездочки. Операнд ALL указывает, что замена имени MASSIV на имя ARRAY должна быть выполнена во всех строках с номерами 60—200. Если операнд ALL отсутствует, замена выполняется только для первой обнаруженной последовательности символов.

Пример 2. В строках с номерами из диапазона 40—100 необходимо изменить последовательность символов, начиная с 31-го символа каждой строки. Этим требованиям удовлетворяет следующая подкоманда CHANGE:

```
CHANGE 40 100 30
```

Для каждой строки с номером из диапазона 40—100 на АП выводятся первые 30 символов. После вывода части строки абонент дополняет ее новой информацией, а затем вводит нажатием клавишного переключателя ВВОД.

3.4. ПРОВЕРКА СИНТАКСИСА ПРЕДЛОЖЕНИЙ

В СРВ имеются средства для выполнения синтаксической проверки исходной программы до ее трансляции. Проверка синтаксиса каждого предложения производится вне связи его с другими предложениями. Поэтому такие ошибки, как дублирующие имена, неопределенные метки и т. д., не обнаруживаются.

Проверка синтаксиса предложений выполняется специальной программой — синтаксическим анализатором, входящим в состав операционной системы. Для вызова синтаксического анализатора существуют два способа: можно либо указать операнд SCAN в команде EDIT, либо использовать подкоманду SCAN команды EDIT.

При задании операнда SCAN синтаксическая проверка выполняется для всех предложений, помещаемых в набор данных в режиме ввода команды EDIT.

Подкоманда SCAN дает возможность более гибко управлять механизмом включения и выключения проверки. В отличие от операнда SCAN подкоманда SCAN позволяет организовать проверку синтаксиса не только вводимых предложений, но и уже записанных в на-

бор данных ранее. С помощью подкоманды SCAN можно выполнить проверку некоторого участка программы, задав границы этого участка в виде номеров строк. Например, по подкоманде SCAN 100 250 производится проверка предложений в строках из диапазона 100—250. Подкоманда SCAN без операндов является запросом на выполнение проверки всех предложений набора данных. Так, чтобы выполнить проверку уже существующего набора данных, достаточно ввести только имя подкоманды SCAN. Чтобы включить проверку предложений, поступающих в набор данных, подкоманду SCAN следует записать в виде SCAN ON. В этом случае все последующие предложения, записываемые в набор данных в режиме ввода, будут проверяться на наличие ошибок. Отменить действие синтаксического анализатора можно вводом подкоманды SCAN OFF. Начиная с этой точки, проверка вводимых строк выполняться не будет. Использование подкоманды SCAN является единственным способом, позволяющим выполнить проверку отдельных предложений, введенных в режиме редактирования команды EDIT, так как операнд SCAN действует только при работе в режиме ввода.

Ошибки в предложениях Фортрана, обнаруженные синтаксическим анализатором, идентифицируются сообщениями с префиксом IPD системы. Сообщения следуют за ошибочным предложением. После вывода сообщений, относящихся к одному предложению, устанавливается режим редактирования (выводится сообщение EDIT) и управление передается абоненту, чтобы он мог ввести правильное предложение. Абонент может продолжить ввод программы, установив предварительно режим ввода команды EDIT.

Пример. Иллюстрируются некоторые способы управления режимом SCAN.

Шаг 1. Ввод команды EDIT с операндом SCAN для работы с набором данных IVANOV.MYPROG.FORT, содержащим программу на Фортране.

```
edit myprog old fortse scan
EDIT
```

Шаг 2. Ввод подкоманды SCAN для проверки синтаксиса всех предложений программы, записанных ранее в набор данных IVANOV.MYPROG.FORT. В результате проверки в строке с номером 270 обнаружена ошибка, на что указывает сообщение IPD026.

```
scan
IPD026 270 *, 5) A DATA SET REF NUMBER EXPECTED
EDIT
```

Шаг 3. Корректировка оператора в режиме редактирования.

```
270 write (5, *) a
```

Шаг 4. Ввод подкоманды SCAN для проверки синтаксиса скорректированного оператора. В операторе ошибок не обнаружено.

```
scan 270
```

Шаг 5. Установка режима ввода с тем, чтобы продолжить запись программы в набор данных. Предложения добавляются в конец набора данных, начиная со строки с номером 540.

```
input
INPUT
00540
```

Шаг 6. Так как в команде EDIT был указан операнд SCAN, то последующие вводимые строки, начиная с номера 540, будут проверяться на наличие ошибок.

```
00540  a=b
00550  go to 1
  . . .
00610  write (6,10) a
00620  (пустая строка)
EDIT
```

Шаг 7. Отмена режима синтаксической проверки.

```
scan off
```

Шаг 8. Выполнение некоторых действий, запланированных абонентом, после которых ему потребовалось продолжить ввод программы. Так как режим проверки был отменен на предыдущем шаге, абонент восстанавливает его с помощью подкоманды SCAN ON.

```
scan on
input
INPUT
00620          call m(1,)
IPD046 620 )     ARGUMENT EXPECTED
EDIT
620            call m(1,3)
scan 620
input
  . . .
00730          end
00740 (пустая строка)
EDIT
```

Шаг 9. Сохранение программы в наборе данных и завершение действий по команде EDIT.

```
save
SAVED
end
READY
```

Глава 4

ОБСЛУЖИВАНИЕ НАБОРОВ ДАННЫХ

В системе разделения времени всю информацию (исходные программы, данные для выполнения программ) абонент хранит в наборах данных на дисках. Сервисные функции по обслуживанию этих наборов данных в СРВ аналогичны тем, которые обеспечиваются утилитами в режиме пакетной обработки. Эти функции запрашиваются с помощью специальных команд, позволяющих распределять, освобождать, переименовывать, удалять наборы данных, получать о них справочную информацию.

4.1. РАСПРЕДЕЛЕНИЕ И ОСВОБОЖДЕНИЕ НАБОРОВ ДАННЫХ

Трансляторы Фортран SE и Фортран CC, а также рабочие программы, создаваемые этими трансляторами, используют входные и выходные наборы данных. Для работы в операционной системе наборы данных необходимо описать, чтобы передать системе основные сведения о них. В пакетном режиме для описания наборов данных предназначается оператор DD. В СРВ описание наборов данных может быть выполнено по оператору DD в процедуре LOGON или по команде ALLOCATE. Такой способ описания называется распределением.

Распределение наборов данных производится постоянно или динамически. Постоянно распределяются наборы данных, с которыми абонент работает в течение всего сеанса. Для каждого такого набора данных в процедуру LOGON включается оператор описания DD. Наборы данных, используемые в сеансе недлительно, рекомендуется распределять динамически до того, как обратиться к ним по командам СРВ. В команде ALLOCATE абонент сообщает операционной системе сведения о наборе данных, такие, как имя набора данных, диспозиция, требования к памяти на дисках. На

основании этой информации система строит управляющие блоки, аналогичные тем, которые создаются по оператору DD. Некоторые команды СРВ сами выдают запросы на динамическое распределение требуемых наборов данных. Например, при выполнении команды FORTSE распределяются наборы данных транслятора Фортран SE: входной набор данных, содержащий исходную программу, и наборы данных для вывода результатов трансляции. В пакетном режиме эти наборы данных определяются операторами DD с именами SYSIN, SYSPRINT, SYSLIN. Наборы данных, необходимые Редактору и Загрузчику, распределяются соответствующими командами LINK и LOADGO.

Максимальное количество наборов данных, распределяемых динамически в сеансе работы, определяется числом блоков управления данными, зарезервированных в процедуре LOGON по операторам DD с операндом DYNAM. В каждый момент абонент может работать с ограниченным количеством наборов данных, не превышающим количество операторов DD в процедуре LOGON. Общее количество наборов данных, используемых на протяжении всего сеанса работы, может быть значительно больше. Такая возможность обеспечивается механизмом динамического распределения и освобождения наборов данных, реализуемого командами языка СРВ ALLOCATE и FREE. В отличие от постоянного распределения, при котором набор данных закрепляется за абонентом на весь сеанс работы, динамическое распределение позволяет абоненту гибко управлять ресурсами вычислительной системы. Набор данных, распределенный абонентом динамически, делается доступным ему только в течение некоторого интервала времени (от момента распределения, например, по команде ALLOCATE, до момента освобождения, например, по команде FREE) и недоступным другим абонентам. Вне этого интервала набор данных может использоваться в параллельных сеансах работы. Кроме того, при динамическом распределении более экономно расходуется основная память для управляющих блоков. Если при постоянном распределении для каждого набора данных строится отдельный управляющий блок, то при динамическом распределении благодаря механизму освобождения один и тот же блок может выделяться нескольким наборам данных.

Основным средством динамического распределения наборов данных на дисках является команда ALLOCATE. Команда ALLOCATE имеет формат:

```
{ ALLOCATE }{ FILE(имя) [ DATASET ({*          }) ] }
{ ALLOC    }{          { имя набора данных }) ] }
           { DATASET ({*          }) [FILE(имя)] }
           { ({имя набора данных }) }

           { NEW    }
           { OLD    }
           { SMR    }  [VOLUME(регистрационный номер)]
           { MOD    }  [SPACE(количество[,приращение])
           { SYSOUT }  BLOCK(длина) [DIR(число)]]
```

Все операнды в команде ALLOCATE — ключевые.

Набор данных идентифицируется с помощью операндов FILE и DATASET. Наличие одного из этих операндов является обязательным. Операнд FILE определяет имя оператора DD, соответствующее набору данных. При распределении наборов данных рабочих программ Фортрана этот операнд необходимо всегда включать в команду ALLOCATE. Операнд DATASET определяет имя набора данных. Если имя набора данных вводится без ограничительных апострофов, описательный уточнитель опускать нельзя: для работы с исходными программами указывается уточнитель FORT, для наборов данных рабочей программы — DATA. Вместо имени набора данных можно указать символ «*», тогда для ввода и вывода информации назначается АП. Для временного набора данных операнд DATASET можно опустить.

Операнды NEW, OLD, SHR и MOD определяют состояние набора данных перед началом выполнения команды. В команде ALLOCATE записывается один из этих операндов. Операнд NEW запрашивает распределение нового набора данных. Для нового набора данных необходимо указать также характеристики устройства прямого доступа, на котором он будет размещаться. В результате распределения набору данных выделяется место на дисках. Если имя набора данных было указано в операнде DATASET, оно заносится в каталог системы. Операнд OLD указывает, что набор данных существует и абонент запрашивает монопольный доступ к нему. Чтобы разрешить использование набора данных несколькими абонентами, вместо OLD следует указать операнд SHR. Операнд MOD обеспечи-

вает добавление записей в набор данных, например, в тех случаях, когда набор данных создается в течение нескольких сеансов работы. При первом обращении к набору данных задается операнд NEW, при всех последующих — операнд MOD.

Для распределения набора данных на личном томе в команду включается операнд VOLUME, в котором указывается регистрационный номер тома, например VOLUME(FORTRN). Если набор данных создается на любом томе, доступном системе, или набор данных уже существует и каталогизирован, то этот операнд можно опустить.

При распределении нового набора данных на дисках нужно указать также требуемый объем памяти, так как на одном томе обычно размещается несколько наборов данных. Для указания объема памяти служат операнды SPACE, BLOCK и DIR. Операнд SPACE определяет объем памяти в блоках. Например, если исходная программа состоит из 500 предложений в стандартном формате, то этот операнд можно записать в виде SPACE(100,1). Для вторичного распределения указан один блок. Размер блока задается в операнде BLOCK в байтах. Максимальная длина блока для наборов данных с исходными программами, подготовленными в стандартном формате, равна 400 байтам, а для программ в свободном формате — 1680 байтам. Для наборов данных рабочих программ размер блока не должен превышать 1680 байт. Размер участка, отводимого набору данных на устройстве прямого доступа, определяется исходя из произведения длины блока на количество блоков. Значение «приращение» используется для вторичного выделения памяти, если первоначальное значение окажется недостаточным. Операнд DIR задается только при распределении нового библиотечного набора данных и указывает количество блоков по 256 байт под оглавление библиотеки.

Операнд SYSOUT служит для определения тех наборов данных, которые направляются в выходной поток. Данные первоначально записываются на устройство прямого доступа, а затем выводятся на печатающее устройство. Выходным классом, которому назначаются такие наборы данных, является класс A.

Команда ALLOCATE широко используется на всех этапах разработки программы в СРВ. На этапе созда-

ния набора данных средствами команды EDIT память для набора данных выделяется на любом томе, доступном системе. Чтобы записать данные на личный том, необходимо ввести команду ALLOCATE, включив в нее операнды DATASET и VOLUME. В операнде DATASET записывается то же имя набора данных, что и в последующей команде EDIT, используемой для создания этого набора данных. В следующем примере создается библиотечный набор данных с именем BIBLIO на томе с регистрационным номером FORTRN. В раздел PART1 этого набора данных записывается исходная программа, содержащая предложения в стандартном формате. Предложения компонуются в блоки размером по 240 байт. Выполнение указанных действий обеспечивают команды:

```
ALLOCATE DATASET('BIBLIO') NEW VOLUME(FORTRN)
         SPACE(50,2) BLOCK(240) DIR(2)
EDIT     'BIBLIO(PART1)' NEW FORTSE SCAN
```

По команде ALLOCATE на томе с регистрационным номером FORTRN для набора данных BIBLIO выделяется память (50 блоков по 240 байт) и выполняется его каталогизация в системе. По команде EDIT программа записывается в раздел PART1 набора данных BIBLIO. Поскольку этот раздел новый, в команде EDIT указан операнд NEW. Для создания в наборе данных BIBLIO другого раздела, например PART2, команда ALLOCATE не требуется, так как набор данных уже каталогизирован:

```
EDIT 'BIBLIO(PART2)' NEW FORTSE SCAN
```

При выполнении рабочих программ, создаваемых трансляторами Фортран SE и Фортран CC, используются различные наборы данных. Имена операторов DD для этих наборов данных в системе зафиксированы. Они имеют формат FTxxFyyy, где xx — номер набора данных, указанный в исходной программе, yyy — порядковый номер набора данных. При распределении наборов данных рабочей программы имя оператора DD задается в операнде FILE команды ALLOCATE.

Следующие примеры иллюстрируют возможности команды ALLOCATE для распределения наборов данных рабочей программы.

Пример 1. Абоненту с идентификатором USER требуется распределить существующий набор данных с име-

нем USER.TEXT.DATA для использования его в рабочей программе. Обращение к набору данных в операторах ввода-вывода производится по номеру 10. Распределение набора данных можно выполнить следующей командой:

```
ALLOCATE FILE(FT10F001) DATASET(TEXT.DATA) OLD
```

Операнд OLD означает, что набор данных существует, и абонент требует его монопольного использования. Если набор данных создан вне СРВ и не каталогизирован, в команду необходимо также включить операнд VOLUME для определения тома, где расположен набор данных:

```
ALLOCATE FILE(FT10F001) DATASET(TEXT.DATA) OLD
    VOLUME(FORTRN)
```

Пример 2. Необходимо распределить новый набор данных с именем OUTPUT на томе с регистрационным номером AAAAAA. Набор данных предназначен для вывода результатов выполнения рабочей программы. Обращение к набору данных в программе производится по номеру 9. Известно, что данные выводятся блоками по 820 байт. Количество запрашиваемых блоков для первичного распределения 20, для вторичного — 2. Указанным условиям удовлетворяет команда:

```
ALLOCATE FILE(FT09F001) DATASET('OUTPUT') NEW
    VOLUME(AAAAAA)
        BLOCK(820) SPACE(20,2)
```

Набор данных создается и каталогизируется в системе. При обращениях к этому набору данных в последующих сеансах работы достаточно указывать только операнды FILE, DATASET и OLD.

Пример 3. Требуется распределить новый временный набор данных, обращение к которому в исходной программе производится по номеру 1. Это можно выполнить командой

```
ALLOCATE FILE(FT01F001) NEW SPACE(10,2) BLOCK(400)
```

Для нового набора данных указан размер памяти на дисках. Так как операнд DATASET отсутствует, набор данных не каталогизируется. Временные наборы данных доступны только в рамках сеанса работы, в котором они создаются. Передавать такие наборы из одного сеанса в другой нельзя.

Пример 4. Необходимо в качестве устройства вывода данных использовать АП. Обращение к набору данных в программе выполняется по номеру 6. Команда ALLOCATE будет иметь следующий вид:

```
ALLOCATE FILE(FT06F001) DATASET(*)
```

Наборы данных, распределенные по команде LOGON или ALLOCATE, не освобождаются автоматически после их использования. Освобождение наборов данных выполняется с помощью команды FREE. Наборы данных идентифицируются в команде FREE своими именами или именами соответствующих операторов DD. В первом случае задается ключевой операнд DATASET, за которым следует список имен одного или нескольких наборов данных или разделов, заключенных в скобки и разделенных пробелами или запятыми. Во втором случае задается ключевой операнд FILE вместе со списком имен операторов DD, связанных с наборами данных, которые должны быть освобождены.

Пример 5.

```
FREE DATASET(PROGM.FORT. BIBLIO.FORT(SUB))
FREE FILE(FT01F001 FT03F001)
```

По первой команде освобождается один последовательный набор данных и один раздел библиотечного набора данных. Если абонент работает с идентификатором TERM, то полные уточненные имена наборов данных имеют вид: TERM.PROGM.FORT и TERM.BIBLIO. FORT(SUB). По второй команде освобождаются два набора данных, связанных с операторами DD с именами FT01F001 и FT03F001.

4.2. УДАЛЕНИЕ НАБОРОВ ДАННЫХ

Наборы данных абонента хранятся на личных томах или томах общего пользования и занимают отведенную им на дисках память. Ресурсы вычислительной системы ограничены. По этой причине наборы данных с истекшим сроком хранения или не используемые абонентом должны удаляться. В результате удаления набор данных перестанет существовать, а запись о нем в каталоге системы стирается. При удалении раздела его имя удаляется из оглавления библиотеки.

С помощью команды DELETE можно удалить один или несколько наборов данных или разделов библиотечных наборов данных.

Команда DELETE содержит один позиционный и один ключевой операнд PURGE или NOPURGE. Позиционный операнд не может быть опущен, он указывает список имен наборов данных или разделов. Имена наборов данных задаются с описательными уточнителями, если они вводятся без ограничительных апострофов. Список, содержащий несколько имен, должен заключаться в круглые скобки. Имена в списке разделяются пробелами или запятыми. Например, чтобы удалить два набора данных USER.PROGM.FORT и USER.TEXT.DATA и раздел SUB из библиотечного набора данных USER.BIBL.FORT, принадлежащих абоненту с идентификатором USER, список имен в команде DELETE можно записать в виде:

```
(PROGM.FORT,TEXT.DATA,BIBL.FORT(SUB))
```

Ключевой операнд PURGE запрашивает удаление наборов данных с неистекшим сроком хранения. По умолчанию принимается NOPURGE.

4.3. ПЕРЕИМЕНОВАНИЕ НАБОРОВ ДАННЫХ

Изменить имя набора данных или имя раздела библиотечного набора данных можно по команде СРВ RENAME. В качестве операндов команды задаются новое и старое имена наборов данных или разделов. Имена наборов данных должны указываться вместе с описательными уточнителями, если они вводятся без ограничительных апострофов.

Пример.

```
RENAME SUB1.FORT FUNC.FORT
RENAME BIBL.FORT(SUB1) BIBL.FORT(SUB2)
```

Первая команда предписывает заменить имя набора данных USER.SUB1.FORT новым именем USER.FUNC.FORT (USER — идентификатор абонента). Вторая команда предписывает изменить в наборе данных USER.BIBL.FORT имя раздела SUB1 на SUB2.

4.4. ПОЛУЧЕНИЕ СПРАВОЧНОЙ ИНФОРМАЦИИ О НАБОРАХ ДАННЫХ

Если абонент испытывает затруднения при использовании своих наборов данных, он может получить о них справочную информацию по командам языка СРВ LISTCAT,LISTALC и LISTDS.

Команда LISTCAT позволяет абоненту запросить имена всех каталогизированных наборов данных, у которых первое поле имени совпадает с идентификатором данного абонента. Команда LISTALC обеспечивает абонента списком имен наборов данных, распределенных для него в сеансе работы постоянно или динамически. Команда LISTDS позволяет выяснить характеристики отдельных наборов данных, каталогизированных или распределенных для абонента. Содержание выводимой на АП информации уточняется с помощью ключевых операндов. Описание допустимых для каждой команды операндов приведено в табл. 6.

Пример 1.

```
LISTALC
LISTALC STATUS HISTORY MEMBERS SYSNAMES
```

Первая команда запрашивает вывод имен всех наборов данных, распределенных для абонента. Вторая команда позволяет получить полную информацию о всех наборах данных, включая временные, распределенных для абонента.

Пример 2.

```
LISTDS PROGM.FORT
```

По этой команде будет выведена информация об основных характеристиках набора данных с именем USE.PROGM.FORT (USE — идентификатор абонента): формат записи, размеры записи и блока, организация набора данных, регистрационный номер тома, на котором размещен набор данных. Для получения дополнительной информации о наборе данных в команду следует включить операнды STATUS, HISTORY, MEMBERS и LABEL.

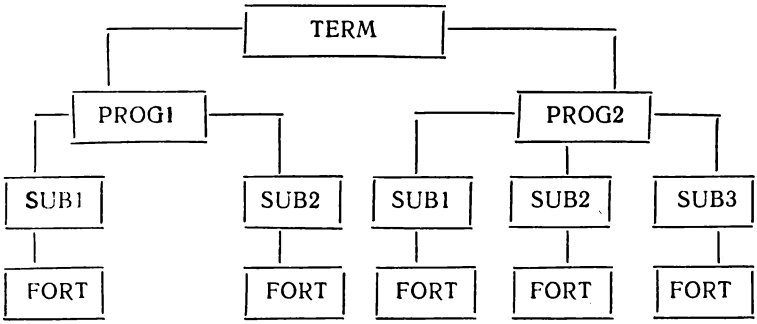
Пример 3. Пусть в системном каталоге имеется информация о наборах данных с именами:

```
TERM.PROG1.SUB1.FORT      TERM.PROG2.SUB1.FORT
TERM.PROG1.SUB2.FORT      TERM.PROG2.SUB2.FORT
                          TERM.PROG2.SUB3.FORT
```

Таблица 6

Операнд	Запрашиваемая информация	Использование в команде		
		LISTCAT	LISTALC	LISTDS
STATUS	Имя оператора DD. Диспозиция набора данных в случае нормального и аварийного завершения программы	Нет	Да	Да
HISTORY	Сведения о наборе данных: дата создания и истечения срока хранения, организация набора данных, отметка о наличии защиты с помощью пароля	Да	Да	Да
MEMBERS	Имена разделов библиотечного набора данных	Да	Да	Да
SYSNAMES	Имена распределенных временных наборов данных	Нет	Да	Нет
VOLUMES	Регистрационные номера томов, на которых размещены каталогизированные наборы данных	Да	Нет	Нет
LEVEL (индекс)	Идентификация группы каталогизированных наборов данных в соответствии со структурой каталога	Да	Нет	Нет
LABEL	Содержимое блоков управления наборами данных (DSCB)	Нет	Нет	Да
(список имен наборов данных)	Список имен наборов данных, о которых запрашивается информация. Является первым позиционным операндом команды	Нет	Нет	Да

Эти имена образуют в системном каталоге следующую индексную структуру:



Необходимо получить полную информацию о наборах данных с именами TERM.PROG1.SUB1.FORT и TERM.PROG1.SUB2.FORT. Для этого в команду LISTCAT следует включить операнд LEVEL(TERM.PROG1) для идентификации группы каталогизированных наборов данных и операнды HISTORY, VOLUMES и MEMBERS.

Раздел III
ОБРАБОТКА ПРОГРАММ НА ФОРТРАНЕ

Глава 5
ОБРАБОТКА ПРОГРАММ НА ФОРТРАНЕ В СРВ

5.1. ОБРАБОТКА ПРОГРАММ ТРАНСЛЯТОРОМ ФОРТРАН СС

В режиме разделения времени транслятор Фортран СС обеспечивает полный цикл обработки программ. С точки зрения пользователя Фортрана обработка программы, начиная с ее трансляции и кончая получением результатов счета, выполняется транслятором как единый непрерывный процесс. Для редактирования и последующего выполнения программы транслятор вызывает Загрузчик ОС ЕС. При редактировании к программе присоединяются необходимые для ее выполнения программы библиотеки Фортрана.

Ориентированный на использование в режиме разделения времени транслятор Фортран СС позволяет: вводить исходную программу в стандартном или свободном формате; выводить на АП диагностические сообщения об ошибках в исходной программе, управляя при этом форматом диагностических сообщений; создавать объектный модуль для диалоговой отладки; выполнять программу и получать результаты счета.

Используя транслятор в СРВ, нельзя вывести распечатку исходной программы, ее можно получить только в режиме пакетной обработки.

Полный цикл обработки программы, включающий ее выполнение, транслятор Фортран СС обеспечивает в том случае, когда программа синтаксически верная или допущенные в ней ошибки не грубые (с кодом серьезности не выше 4). Иначе выполняется только трансляция. Редактирование и счет по программе отменяются. Диагностические сообщения об ошибках, обнаруженных во время трансляции, выводятся на АП в сжа-

том или расширенном формате. Описание сообщений транслятора приводится в приложении 1. Синтаксически правильная программа, в которой допущены логические ошибки, должна пройти этап отладки. Средства диалоговой отладки обсуждаются в гл. 7.

5.1.1. Трансляция, редактирование и выполнение

Вызов транслятора Фортран СС в СРВ производится по команде FORTCC, имеющей формат:

FORTCC имя набора данных [FIXED / FREE] [LMSG / SMSG] [TEST]

Первый операнд — позиционный. Он указывает имя входного набора данных, содержащего исходную программу. Исходная программа может состоять из одного или нескольких программных модулей. Все предложения в программных модулях готовятся в едином формате: стандартном или свободном.

Операнды FIXED и FREE задают формат представления предложений исходной программы: FIXED означает, что предложения подготовлены в стандартном формате, FREE — в свободном формате.

Операнды LMSG и SMSG позволяют управлять форматом диагностических сообщений транслятора. Если указать операнд SMSG, то сообщения об ошибках в программе будут выведены в сжатом формате. Определив операнд LMSG, абонент получит сообщения в расширенном формате, в которых причина ошибок будет диагностирована полнее, чем в случае сжатого формата.

Операнд TEST устанавливает режим вывода объектного модуля в набор данных в виде, пригодном для последующей диалоговой отладки. При задании операнда TEST в объектном модуле строится дополнительная информация, обеспечивающая связь с Диалоговым отладчиком. Объектный модуль (или объектные модули, если программа состояла из нескольких программных модулей) затем может использоваться Диалоговым отладчиком в качестве объекта отладки. Построенный объектный модуль сохраняется в наборе данных, имя которого генерируется из имени входного набора данных по следующим правилам. Если имя на-

бора данных было указано в команде FORTCC без ограничительных апострофов, например PROG (этому имени соответствует полное уточненное имя USER. PROG.FORT, где USER — идентификатор абонента), то имя выходного набора данных строится путем замены описательного уточнителя FORT на OBJ. В данном примере выходной набор данных сохранится под именем USER.PROG.OBJ. Если имя входного набора данных было введено полностью, т. е. в апострофах, например 'SOURCE', то выходной набор данных получит имя USER.SOURCE.OBJ (к указанному в команде имени достраиваются идентифицирующий уточнитель USER и описательный уточнитель OBJ).

По команде FORTCC посредник транслятора выполняет подготовительную работу перед вызовом транслятора: автоматически распределяет необходимые транслятору и Загрузчику наборы данных и устанавливает режимы трансляции в зависимости от указанных в команде операндов. Входной набор данных, содержащий исходную программу, распределяется всегда. Если абонент опустил его, выдается подсказывающее сообщение с требованием ввести имя этого набора данных. Сообщения посредника приведены в приложении 1. После ввода абонентом имени набора данных выполнение команды FORTCC продолжается. Кроме набора данных с исходной программой, по команде FORTCC распределяется набор данных SYS1.FORTLIB, содержащий библиотеку программ Фортрана, и наборы данных рабочей программы, обращение к которым в программе производится по номерам 5 и 6 (последние распределяются на АП). Если в команде FORTCC был указан операнд TEST, то распределяется также набор данных для вывода объектного модуля.

В команде FORTCC нет операндов, позволяющих определить личные библиотеки объектных или загрузочных модулей. Поэтому все программные модули, необходимые для выполнения головного модуля, предварительно должны быть добавлены в библиотеку SYS1.FORTLIB в формате загрузочных модулей или помещены во входной набор данных в исходном виде вместе с головным модулем.

По команде FORTCC запрашивается полный цикл обработки программы (трансляция, редактирование и выполнение). Поэтому абонент должен заранее поза-

ботиться о распределении входных и выходных наборов данных рабочей программы. Распределение наборов данных может быть выполнено постоянно (в процедуре LOGON) или динамически (по команде ALLOCATE). Такое распределение не требуется, если данные вводятся из набора данных с номером 5, а результаты вычислений посылаются в набор данных с номером 6 и в качестве устройства ввода и вывода назначен АП.

Запросить полный цикл обработки программы транслятором Фортран СС можно также с помощью команды RUN или подкоманды RUN команды EDIT. Команда RUN позволяет указать те же режимы трансляции, что и команда FORTCC, за исключением режима TEST. Поэтому если в дальнейшем предполагается диалоговая отладка программы, то для получения объектного модуля трансляцию следует выполнить по команде FORTCC.

В подкоманде RUN указывается только формат диагностических сообщений. Имя и тип набора данных с исходной программой, а также формат предложений Фортрана определяются соответствующей командой EDIT.

Команда RUN и подкоманда RUN имеют соответственно форматы:

RUN имя набора данных FORTCC [FIXED / FREE] [LMSG / SMSG]

RUN [LMSG / SMSG]

Пример. Необходимо протранслировать, отредактировать и выполнить программу, которая находится в разделе SUB1 библиотечного набора данных с именем USER.SOURCE.FORT. Программа подготовлена в свободном формате, диагностические сообщения транслятора запрашиваются в расширенном формате. Идентификатор абонента — USER. Указанные действия могут быть выполнены командой FORTCC SOURCE(SUB1) LMSG или RUN SOURCE(SUB1) FORTCC LMSG.

5.1.2. Сеанс работы с транслятором Фортран СС

Полный сеанс работы, в котором иллюстрируются возможности обработки программ с помощью транслятора Фортран СС, рассматривается на примере про-

граммы вычисления корней квадратного уравнения вида $ax^2+bx+c=0$. Программа состоит из головного модуля и подпрограммы. В головном модуле производится ввод с АП коэффициентов уравнения и вывод на АП значений корней уравнения. В подпрограмме вычисляются корни уравнения для заданных значений коэффициентов. Сеанс работы состоит из 17 шагов и отражает основные этапы обработки программы в СРВ.

Шаг 1. Определение начала сеанса работы. Сеанс работы начинается вводом команды LOGON. Эта команда предназначена для идентификации абонента в СРВ. В команде LOGON абонент указывает свой идентификатор IVANOV и имя FORTR процедуры LOGON. Пароль и учетный номер абонент не использует.

С помощью сообщения NO BROADCAST MESSAGES система информирует абонента о том, что в его адрес до начала сеанса работы не поступили сообщения от оператора ЭВМ и других абонентов. Пока выполняется обработка команды LOGON, СРВ периодически посылает абоненту сообщение LOGON PROCEEDING. После завершения обработки команды LOGON выводится сообщение READY, означающее, что связь абонента с системой установлена и можно вводить любые команды СРВ.

Шаг 2. Определение характеристик сеанса работы. Для задания характеристик сеанса работы используются команды TERMINAL и PROFILE. В команде TERMINAL операнд INPUT определяет символ %, имитирующий переключатель ВНИМАНИЕ. Для вызова прерывания по сигналу ВНИМАНИЕ абонент должен ввести символ % как только получит разрешение на ввод с АП. В команде PROFILE задаются два операнда. Операнд MSGID указывает, что все последующие сообщения должны выводиться с идентификаторами. Идентификаторы позволяют легко установить компонент операционной системы, который выдал соответствующее сообщение. Операнд PROMPT задает режим вывода подсказывающих сообщений. В случае ошибок, допущенных при вводе команд СРВ, абоненту будет предоставлена возможность исправить ошибку и продолжить выполнение команды. Обычно этот режим устанавливается по умолчанию.

Шаг 3. Ввод команды EDIT для записи исходной программы. Запись исходной программы будет выполняться в новый набор данных (операнд NEW). Полное уточненное имя набора данных IVANOV.PROGM.FORT строится по правилам, действующим в СРВ: IVANOV — идентификатор абонента (указан в команде LOGON), PROGM — имя набора данных (указано в команде EDIT), FORT — описательный уточнитель (определяется на основании типа данных FORTCC из команды EDIT). Набор данных с исходной программой будет размещаться на томе прямого доступа общего пользования. Операнд FORTCC в команде EDIT указывает, что программа вводится в свободном формате (режим FREE принимается по умолчанию) для последующей обработки транслятором Фортран СС. Последний операнд SCAN требует, чтобы при вводе предложений Фортрана была выполнена их синтаксическая проверка. Операнд NUM, задающий нумерацию строк, явно не указан, он предполагается по умолчанию. Так как

набор данных IVANOV.PROGM.FORT новый, устанавливается режим ввода. После получения сообщения INPUT абонент может начать ввод исходной программы.

Шаг 4. Запись головного модуля в набор данных. Нумерация строк начинается с 10 и выполняется с шагом 10. Первые две строки введены правильно. В строке с номером 30 допущена синтаксическая ошибка, на которую указывает диагностическое сообщение IPD012. Ввод программы прекращается, и устанавливается режим редактирования. Абоненту предоставляется возможность скорректировать ошибочное предложение.

Шаг 5. Корректировка предложения. Для корректировки предложения достаточно набрать номер его строки и через пробел правильный текст. Так как предложения Фортрана, введенные в режиме редактирования, не контролируются на наличие синтаксических ошибок, то последующая проверка этого предложения выполняется по подкоманде SCAN.

Шаг 6. Продолжение записи головного модуля. Чтобы продолжить запись программы, необходимо вернуться в режим ввода команды EDIT. Для этого абонент вводит пустую строку. Нумерация последующих строк начинается с номера 40.

Шаг 7. Запись в набор данных подпрограммы SOLVE. Производится ввод подпрограммы SOLVE. После его завершения для перехода в режим редактирования абонент вводит пустую строку.

Шаг 8. Вывод программы на АП. Чтобы визуально проверить программу, абонент выводит ее текст на АП по подкоманде LIST. При анализе программы абонент обнаруживает, что в подпрограмме SOLVE пропущены три предложения после строки 220.

Шаг 9. Корректировка программы. Корректировка программы производится в режиме ввода команды EDIT, который устанавливается по подкоманде INPUT. Операнды подкоманды INPUT указывают, что строкам должны быть присвоены номера 222, 224 и 226. После ввода правильного текста, чтобы закончить корректировку программы, абонент вводит пустую строку. Устанавливается режим редактирования.

Шаг 10. Сохранение программы. Абонент вначале перенумеровывает строки набора данных с шагом 10, а затем сохраняет его под именем IVANOV.PROGM.FORT.

Шаг 11. Трансляция программы. Для выполнения полного цикла обработки программы транслятором Фортран CC абонент вводит подкоманду RUN команды EDIT. Операнд LMSG в подкоманде RUN задает вывод диагностических сообщений транслятора в расширенном формате. При трансляции в подпрограмме SOLVE обнаружена ошибка: оператор в строке с номером 90 передает управление оператору с меткой 6, которая не определена в подпрограмме. Транслятор выдает два сообщения, за которыми следует неопределенная метка. Из-за ошибки в программе ее дальнейшая обработка (редактирование и выполнение) прекращается и устанавливается режим редактирования команды EDIT.

Шаг 12. Корректировка программы после трансляции. Абонент корректирует строку с номером 150, проверяет ее синтаксис (по подкоманде SCAN) и затем сохраняет измененный набор данных (по подкоманде SAVE).

Шаг 13. Повторная трансляция программы. Для выполнения полного цикла обработки программы абонент повторно вводит

подкоманду RUN, но при задании операнда LMSG допускает ошибку: вместо LMSG набирает MSG. Выдается подсказывающее сообщение, запрашивающее ввод верного операнда. Абонент вводит LMSG.

Шаг 14. Ввод коэффициентов уравнения. Отсутствие диагностических сообщений указывает на то, что трансляция программы прошла успешно. Так как исходная программа не содержит ошибок, транслятор автоматически обеспечивает редактирование и выполнение программы. Во время выполнения головного модуля выдается сообщение, содержащее метку оператора READ, запрашивающего ввод данных. В ответ на это сообщение абонент вводит с АП значения коэффициентов a, b и c, равные соответственно 1, 5, 0. Результаты вычислений выводятся на АП. Каждый корень представляется парой чисел. Первое число является действительной частью корня, второе — мнимой. Выполнение программы завершается и устанавливается режим ввода подкоманд команды EDIT.

Шаг 15. Переход в режим ввода команд. По окончании отладки программы абонент вводит подкоманду END и тем самым завершает выполнение команды EDIT.

Шаг 16. Выполнение программы. Так как набор данных с исходной программой создан и отладка ее прошла успешно, выполнение программы можно запросить по команде FORTCC или RUN. Для вычисления корней уравнения при новых значениях коэффициентов a, b и c абонент использует команду FORTCC.

Шаг 17. Завершение сеанса работы. Сеанс работы завершается по команде LOGOFF. СРВ выводит информационное сообщение о времени окончания сеанса работы.

Протокол сеанса работы имеет следующий вид:

```
logon ivanov proc(fortr)                                    Шаг 1
IVANOV LOGON IN PROGRESS AT 17:36:33 ON MARCH 2, 1981
NO BROADCAST MESSAGES
LOGON PROCEEDING
READY
terminal input(%)                                           Шаг 2
READY
profile msgid prompt
READY
edit progm new fortcc scan                                  Шаг 3
INPUT
00010 common x1real,x1imag,x2real,x2imag                    Шаг 4
00020 1read(5,*)a,b,c
00030 call solve(a,b,c
IPD012I    30)    EXPECTED
EDIT
30 call solve(a,b,c)                                        Шаг 5
scan 30
(пустая строка)                                             Шаг 6
INPUT
00040  write(6,*)x1real,x1imag,x2real,x2imag
00050  end
00060  subroutine solve(a,b,c)                              Шаг 7
```

```
00070  common x1real,x1imag,x2real,x2imag
00080  disc=b**2—4.0*a*c
00090  if(disc)5,6,7
00100  5x1real=—b/(2.0*a)
00110  x2real=x1real
00120  x1imag=sqrt(—disc)/(2.0*a)
00130  x2imag=—x1imag
00140  return
00150  x1real=—b/(2.0*a)
00160  x2real=x1real
00170  x1imag=0.
00180  x2imag=0.
00190  return
00200  7s=sqrt(disc)
00210  x1real=(—b+s)/(2.0*a)
00220  x2real=(—b—s)/(2.0*a)
00230  end
00240  (пустая строка)
EDIT
```

Шаг 8

```
list
00010 COMMON  X1REAL,X1IMAG,X2REAL,X2IMAG
00020  1READ(5,*)A,B,C
00030  CALL SOLVE(A,B,C)
00040  WRITE(6.*)X1REAL,X1IMAG,X2REAL,X2 IMAG
00050  END
00060  SUBROUTINE SOLVE(A,B,C)
00070  COMMON X1REAL,X1IMAG,X2REAL,X2REAL,X2IMAG
00080  DISC=B**2—4.0*A*C
00090  IF(DISC)5,6,7
00100  5X1REAL=—B/(2.0*A)
00110  X2REAL=X1REAL
00120  X1IMAG=SQRT(—DISC)/(2.0*A)
00130  X2IMAG=—X1IMAG
00140  RETURN
00150  X1REAL=—B/(2.0*A)
00160  X2REAL=X1REAL
00170  X1IMAG=0.
00180  X2IMAG=0.
00190  RETURN
00200  7S=SQRT(DISC)
00210  X1REAL=(—B+S)/(2.0*A)
00220  X2REAL=(—B—S)/(2.0*A)
00230  END
IKJ52500I END OF DATA
```

Шаг 9

```
input  222 2
00222  x1imag=0.
00224  x2imag=0.
00226  return
00228  (пустая строка)
EDIT
```

Шаг 10

```
renum
save
SAVED
```

```
run lmsg                                                    Шаг 11
IGK420I  00000150 STMT FOLLOWING A TRANSFER OF
         CONTROL HAS NO STMT NMBR
IGK476I  00000260 LABELS IN SOURCE UNDEFINED
     6
EDIT
150 6x1real=—b/(2.0*a)                                      Шаг 12
scan 150
save
SAVED
run msg                                                     Шаг 13
IKJ56712I INVALID KEYWORD, MSG
IKJ56703A REENTER—
lmsg
? 00001                                                     Шаг 14
1,5,0
.0              .0              —5.00000000         .0
EDIT
end                                                         Шаг 15
READY
fortcc progm                                                Шаг 16
? 00001
1.5, 10.3, —15.2
1.24866486          .0          —8.11533070          .0
READY
loqoff                                                      Шаг 17
IKJ56470I IVANOV LOGGED OFF TSO AT 18:15:01
    ON MARCH 2, 1981+
```

5.2. ОБРАБОТКА ПРОГРАММ ТРАНСЛЯТОРОМ ФОРТРАН SE

Обработка программы в СРВ транслятором Фортран SE представляется в виде четырех самостоятельных этапов. Первый этап обработки программы — трансляция, во время которой выявляются синтаксические ошибки в программе. После устранения ошибок трансляция повторяется для получения объектного модуля. Если исходная программа состояла из нескольких программных модулей, то результатом трансляции также будет несколько объектных модулей. Второй этап обработки — редактирование программы. На этом этапе к полученным после трансляции объектным модулям Редактор присоединяет необходимые загрузочные модули программ библиотеки Фортрана, а также загрузочные и объектные модули ранее протранслированных программных модулей. Выходной информацией Редактора является загрузочный модуль. На третьем этапе обработки программы выполняется ее отладка (способы диалоговой отладки программы рас-

сматриваются в гл. 7). Четвертым этапом обработки программы является выполнение загрузочного модуля для получения результатов счета.

Последовательность выполнения этих этапов планируется абонентом в виде одного или нескольких шагов задания. Выполнение каждого шага задания запрашивается следующими командами СРВ: FORTSE — трансляция, LINK — редактирование, CALL — выполнение, LOADGO — редактирование и выполнение, RUN (команда или подкоманда) — трансляция, редактирование и выполнение. Используя эти команды, обработку программы можно представить в виде одной из следующих последовательностей команд:

а) FORTSE
LINK
CALL

б) FORTSE
LOADGO

в) RUN

5.2.1. Трансляция

На этапе трансляции программы можно получить: распечатку исходной программы, диагностические сообщения об ошибках в исходной программе, объектный модуль в наборе данных, распечатку объектного модуля. Вызов транслятора Фортран SE в СРВ производится по команде FORTSE, имеющей формат:

FORTSE имя набора данных
[PRINT [({ * })]] [список режимов]
[NOPRINT [{ имя набора данных }]

Первый операнд является позиционным, он указывает имя входного набора данных, содержащего исходную программу. Исходная программа может состоять из одного или нескольких программных модулей.

Операнд PRINT определяет набор данных печати, в который выводятся распечатки результатов трансляции. Имя набора данных можно опустить. Тогда распечатки результатов трансляции сохраняются в наборе данных, имя которого строится из имени входного набора данных по следующим правилам. Если имя набора данных с исходной программой было указано в команде FORTSE без ограничительных апострофов, например PROG (этому имени соответствует полное уточненное имя USER.PROG.FORT, где USER — идентификатор абонента), то имя набора данных печати строится путем замены описательного уточнителя FORT на

LIST. В данном примере распечатки сохраняются в наборе данных с именем USER.PROG.LIST. Если имя набора данных с исходной программой введено в апострофах, например 'SOURCE(PART)', то набор данных печати получит имя USER.SOURCE.LIST(PART), т. е. к имени, указанному в команде FORTSE, достраиваются идентифицирующий уточнитель USER и описательный уточнитель LIST. Распечатки трансляции могут быть направлены на АП. В этом случае операнд PRINT кодируется в виде PRINT(*). Чтобы отменить вывод распечаток трансляции, в команде FORTSE следует задать операнд NOPRINT.

Список режимов состоит из ключевых операндов, которые определяют режимы трансляции. Операнды записываются в произвольном порядке и отделяются друг от друга запятыми или пробелами. Режимы команды FORTSE позволяют управлять: выводом распечаток трансляции (SOURCE, NOSOURCE, LIST, NOLIST, MAP, NOMAP, LINECNT); выводом объектного модуля (LOAD, NOLOAD, TEST, NOTEST, NAME, ID, NOID); выводом сообщений транслятора на АП (TERM, NOTERM).

Операнд SOURCE задает вывод распечатки исходной программы вместе с сообщениями об ошибках в предложениях Фортрана на АП или в набор данных печати. Операнд NOSOURCE отменяет режим SOURCE.

Операнд MAP задает вывод распечатки распределения памяти для элементов исходной программы на АП или в набор данных печати. Операнд NOMAP отменяет режим MAP.

Операнд LIST задает вывод распечатки объектного модуля в формате команд языка Ассемблера на АП или в набор данных печати. Операнд NOLIST отменяет режим LIST.

Операнд TERM задает вывод диагностических сообщений транслятора на АП. Распечатка содержит неверные предложения Фортрана вместе с сообщениями о допущенных в них ошибках, а также сведения о ходе трансляции. Сообщения об ошибках выводятся по одному в строке за каждым ошибочным предложением. В случае пронумерованного набора данных слева от предложения выводится номер строки. Операнд NOTERM отменяет режим TERM.

Операнд LOAD(имя набора данных) задает вывод объектных модулей для редактирования в указанный набор данных. Имя набора данных в операнде LOAD может быть опущено. В этом случае построенные объектные модули сохраняются в наборе данных, имя которого строится из имени входного набора данных по следующим правилам. Если имя входного набора данных было указано в команде FORTSE без ограничительных апострофов, например PROG (этому имени соответствует полное уточненное имя USER.PROG.FORT, где USER — идентификатор абонента), то имя набора данных с объектными модулями получается заменой описательного уточнителя FORT на OBJ. В данном примере объектные модули сохраняются в наборе данных с именем USER.PROG.OBJ. Если имя входного набора данных введено в апострофах, например 'SOURCE(PART)', то набор данных с объектными модулями получит имя USER.SOURCE.OBJ(PART), т. е. к имени входного набора данных достраиваются идентифицирующий уточнитель USER и описательный уточнитель OBJ. Операнд NOLOAD отменяет режим LOAD.

Операнд TEST устанавливает формат вывода объектных модулей в набор данных для последующей диалоговой отладки. При задании этого режима в каждом объектном модуле строится дополнительная информация, обеспечивающая диалоговую отладку программы на уровне входного языка. Построенные объектные модули сохраняются в наборе данных, имя которого задается операндом LOAD (режим TEST используется вместе с режимом LOAD). Операнд NOTEST отменяет режим TEST.

Операнд NAME(имя программы) указывает имя, присваиваемое головному модулю. Имя может содержать до 6 буквенно-цифровых символов, первый символ должен быть буквой. По умолчанию головной модуль получает имя MAIN. Это же имя должно быть присвоено головному модулю, если планируется диалоговая отладка программы.

Операнд LINECNT(число) указывает максимальное количество строк (но не более 99) на листе при выводе распечаток результатов трансляции.

Операнд ID указывает, что в объектном модуле должны строиться последовательные номера тех операторов программы, которые содержат обращение к

процедурам. Эти номера используются в распечатке списка вызываемых процедур. Список вызываемых процедур отражает последовательность вызова процедур при возникновении ошибки во время выполнения рабочей программы. Операнд NOID отменяет режим ID.

Все распечатки, выдаваемые транслятором Фортран SE по любому из режимов SOURCE, LIST и MAP, направляются в один набор данных, указанный в операнде PRINT. Нельзя, например, вывести распечатку исходной программы на АП, а распечатку объектного модуля сохранить в наборе данных. Диагностическая информация, выводимая по режиму SOURCE, такая же, как и по режиму TERM — отличие состоит только в формате печати диагностических сообщений. При указании режима SOURCE в каждой строке печатается несколько сообщений, при указании режима TERM — по одному сообщению в строке. Описание результатов трансляции, выдаваемых на АП и в набор данных в зависимости от значения операндов PRINT и TERM, содержится в табл. 7. Форматы распечаток трансляции приведены в гл. 6.

Если в команде FORTSE операнд PRINT опущен, то имя набора данных для вывода распечаток трансляции строится из имени набора данных с исходной программой.

По умолчанию команда FORTSE устанавливает следующие режимы трансляции: NOSOURCE, NOMAP, NOLIST, LOAD, TERM, NOTEST, NOID.

На основании информации, содержащейся в команде FORTSE, посредник транслятора Фортран SE производит подготовительные действия перед вызовом транслятора: проверку синтаксиса заданных в команде FORTSE операндов, распределение требуемых для выполнения транслятора наборов данных, построение списка режимов трансляции. Посредник распределяет для транслятора следующие наборы данных: входной набор данных с исходной программой, набор данных для вывода распечаток результатов трансляции, набор данных для вывода объектных модулей, набор данных для вывода диагностических сообщений. Набор данных с исходной программой распределяется всегда. Наборы данных для вывода результатов трансляции распределяются в зависимости от режимов трансляции, указанных в команде FORTSE.

Таблица 7

Операнды		Выводимая информация	
		АП	набор данных
TERM	PRINT	Диагностическая информация	Распечатки трансляции (по режимам SOURCE, MAP, LIST) вместе с диагностической информацией
NOTERM	PRINT	Нет	
TERM	PRINT(*)	Распечатки трансляции (по режимам SOURCE, MAP, LIST) вместе с диагностической информацией	Нет
NOTERM	PRINT(*)		
TERM	PRINT(имя набора данных)	Диагностическая информация	Распечатки трансляции (по режимам SOURCE, MAP, LIST) вместе с диагностической информацией
NOTERM	PRINT(имя набора данных)	Нет	
TERM	NOPRINT	Диагностическая информация	Нет
NOTERM	NOPRINT	Нет	

Если при вводе команды допущены ошибки, выдаются подсказывающие сообщения. Абонент получит подсказывающее сообщение, например, если он не укажет имя набора данных с исходной программой, неверно определит один из режимов трансляции, неверно запишет назначение в операнде NAME или LINECNT. Для продолжения выполнения команды FORTSE необходимо ввести требуемую информацию: имя набора данных, режим и т. д. Некоторые ошибки, обнаруженные во время выполнения команды FORTSE, не могут быть сразу исправлены. Такие ошибки обычно возникают при

распределении наборов данных. Например, если динамически распределено слишком много наборов данных, то запрос на распределение очередного набора данных не будет удовлетворен. В этом случае необходимо вначале освободить неиспользуемые наборы данных, а затем повторно ввести команду FORTSE. Сообщения посредника транслятора Фортран SE приведены в приложении 1.

Пример. Необходимо протранслировать программу, которая находится в наборе данных с именем USER.PROG.FORT, и получить на АП распечатки исходной программы вместе с диагностическими сообщениями об ошибках и распечатки распределения памяти. Идентификатор абонента — USER. Команда будет иметь следующий вид:

FORTSE PROG PRINT (*) SOURCE MAP

Для вывода на АП только диагностической информации об ошибках можно использовать команду FORTSE вида: FORTSE PROG NOLOAD.

5.2.2. Редактирование

Редактирование — следующий после трансляции этап обработки программы. Редактор объединяет объектные модули, полученные в результате трансляции программы, с программами библиотеки Фортрана из набора данных SYS1.FORTLIB и строит загрузочный модуль, который сохраняет в библиотечном наборе данных. На этапе редактирования к программе могут быть подключены объектные модули ранее протранслированных модулей-процедур, а также загрузочные модули из библиотек общего пользования.

Возможности Редактора в СРВ такие же, как и в пакетном режиме ОС. Для вызова Редактора в СРВ используется команда LINK, имеющая формат:

LINK (список имен наборов данных) FORTLIB [LIB(список имен наборов данных)] [LOAD[(имя набора данных)]] [PRINT [({ * / имя набора данных })] / NOPRINT] [список режимов]

Первый операнд является позиционным — он указывает имена входных наборов данных, содержащих объектные и загрузочные модули. Объектные модули могут находиться в последовательных и библиотечных наборах данных, а загрузочные модули — только в библио-

течных наборах данных. Например, пусть программа состоит из трех объектных модулей. Головной модуль находится в последовательном наборе данных USER.MAIN.OBJ, модули-процедуры SUB1 и SUB2 — в разделе SUB набора данных USER.BIBLIO.OBJ.

В этом случае абонент, работающий с идентификатором USER, список в команде LINK может задать в виде: (MAIN, BIBLIO(SUB)). Имена в списке разделяются пробелами или запятыми. Если в списке указано только одно имя, скобки можно опустить.

Ключевой операнд FORTLIB определяет библиотеку автовызова с именем SYS1.FORTLIB, содержащую загрузочные модули программ библиотеки Фортрана. Этот операнд является обязательным в команде LINK. Имена дополнительных личных библиотек загрузочных модулей, используемых при редактировании программы, задаются в операнде LIB. Обычно в этих библиотеках хранятся ранее протранслированные и отредактированные программные модули. Такие наборы данных должны содержать только загрузочные модули.

Имя библиотечного набора данных, в который требуется поместить результат редактирования — загрузочный модуль, указывается в операнде LOAD. Если имя набора данных опущено, то присвоенное ему имя будет сформировано из имени входного набора данных. Например, если объектный модуль находится в наборе данных PROG, то загрузочный модуль сохранится в наборе данных USER.PROG.LOAD (добавляется идентификатор абонента USER и описательный уточнитель LOAD). Если указанному в команде LINK имени PROG соответствует полное уточненное имя входного набора данных USER.PROG.OBJ, то имя набора данных с загрузочным модулем получится заменой описательного уточнителя OBJ на LOAD, в нашем случае будет построено имя USER.PROG.LOAD.

Редактор обеспечивает абонента большим количеством информации, полезной при отладке программы. Эта информация может быть выведена на АП или сохранена в наборе данных. В первом случае необходимо указать операнд PRINT(*), во втором — PRINT (имя набора данных). Содержание выводимой информации, так же как и режимы редактирования, определяются операндами, заданными в списке режимов.

Допустимые в СРВ режимы Редактора такие же, как и при работе в режиме пакетной обработки. Специально следует отметить режим TERM. При задании этого режима в СРВ вывод сообщений об ошибках редактирования будет производиться на АП. Именование наборов данных в команде LINK подчиняется правилам, действующим в СРВ. Описательные уточнители, присваиваемые по умолчанию именам наборов данных, приведены в табл. 3.

Пример 1. Требуется отредактировать объектные модули, которые находятся в наборе данных с именем USE.PROG.OBJ, и поместить загрузочный модуль в раздел A набора данных USE.PROG1.LOAD. Для этого достаточно ввести команду

```
LINK PROG FORTLIB LOAD(PROG1(A))
```

Пример 2. Необходимо выполнить редактирование программы, состоящей из трех объектных модулей, которые находятся в разделе A набора данных с именем PROG. Кроме библиотеки программ Фортрана, используются загрузочные модули из библиотечного набора с именем BIBLIO. Полученный загрузочный модуль требуется поместить в раздел MY набора данных PAKET. Распечатки редактирования, содержащие схему распределения памяти загрузочного модуля и список управляющих операторов Редактора, необходимо сохранить в наборе данных с именем LIST, сообщения об ошибках редактирования направить на АП. Команда LINK может быть записана в виде:

```
LINK 'PROG(A)' FORTLIB...LIB('BIBLIO')
     LOAD('PAKET(MY)') PRINT('LIST') LIST MAP TERM
```

5.2.3. Выполнение

Выполнение — последний этап обработки программы. Чтобы вызвать программу для выполнения необходимо ввести команду CALL, указав в ней имя набора данных с программой. Выполняемая программа всегда представляется в формате загрузочного модуля, который хранится в разделе библиотечного набора данных. Например, чтобы вызвать для выполнения программу из раздела NAME набора данных USE.ZAGR.LOAD, абоненту, работающему с идентификатором USE, достаточно ввести команду CALL ZAGR(NAME). Если имя набора данных указывается

полностью, то оно вводится в ограничительных апострофах, например CALL 'BIBL(PART1)'.

Перед вводом команды CALL следует распределить все используемые в программе наборы данных. Распределение наборов данных может быть выполнено постоянно в процедуре LOGON или динамически по команде ALLOCATE. Обычно наборы данных, обращение к которым производится по номерам 5 и 6, распределяются постоянно на АП в процедуре LOGON. Такое распределение упрощает работу в режиме диалога, так как появляется возможность вводить и выводить на АП данные рабочей программы, включая сообщения об ошибках.

Для облегчения ведения отладки (с целью визуального контроля за ходом выполнения программы) рекомендуется вначале все входные и выходные наборы данных рабочей программы назначать на АП. Например, если обращение к входному и выходному наборам данных производится соответственно по номерам 8 и 9, то их можно распределить на АП следующими командами:

```
ALLOCATE FILE(FT08F001) DATASET(*)
ALLOCATE FILE(FT09F001) DATASET(*)
```

Звездочка в операнде DATASET указывает, что для ввода и вывода данных назначается АП. После того как программа отлажена и требуется ее длительная эксплуатация, для входного набора данных можно указать устройство прямого доступа, а результаты выполнения программы направить в выходной поток, например:

```
ALLOCATE FILE(FT08F001) DATASET(TEXT.DATA) OLD
ALLOCATE FILE(FT09F001) SYSOUT
```

В приведенном примере входные данные находятся в наборе данных с именем USE.TEXT.DATA (USE — идентификатор абонента).

5.2.4. Редактирование и выполнение

Редактирование и выполнение программы можно оформить в виде двух или одного шага задания. В первом случае следует использовать команды LINK и CALL, во втором случае должна использоваться команда LOADGO.

Команда LOADGO вызывает Загрузчик ОС, который создает загрузочный модуль и сразу же его выполняет. Полученный загрузочный модуль в наборе данных на дисках не сохраняется, поэтому каждому выполнению программы должно предшествовать редактирование. Загрузчик, как и Редактор, объединяет объектные и загрузочные модули в единую программу, настраивает ее по месту основной памяти с учетом фактических адресов и передает управление в точку входа полученной программы. Для разрешения внешних ссылок Загрузчик ведет поиск необходимых загрузочных модулей в библиотеке программ Фортрана и в библиотеках, указанных в команде LOADGO.

Команда LOADGO имеет формат:

```
{ LOADGO }  (список имен наборов данных) FORTLIB
{ LOAD   }
            [LIB(список имен наборов данных)]
            [ PRINT [({*                  })]]
            [       [({имя набора данных})]]
            [ NOPRINT                      ]
            [список режимов]
```

Операнды в команде LOADGO используются для тех же целей, что и в команде LINK.

Первый операнд является позиционным — он указывает имена наборов данных, содержащих объектные и загрузочные модули.

Ключевые операнды FORTLIB и LIB предназначаются для описания библиотек загрузочных модулей, используемых при редактировании программы.

Список режимов определяет режимы редактирования и содержимое распечаток, выводимых на АП или в набор данных в соответствии с операндом PRINT. По сравнению с Редактором у Загрузчика меньше средств для задания режимов редактирования.

Именование наборов данных в команде LOADGO удовлетворяет правилам, действующим в СРВ. Описательные уточнители, присваиваемые по умолчанию именам наборов данных, приведены в табл. 3.

Пример. Необходимо отредактировать и выполнить программу, которая состоит из двух объектных модулей, записанных в наборах данных с именами US.MAIN.OBJ и US.SUB.OBJ. Сообщения об ошибках требуется направить на АП. Кроме программ библиотеки Фортрана, используется модуль-процедура из набора данных с именем US.BIBLIO.LOAD. Абонент,

работающий с идентификатором US, может ввести команду:

```
LOAD (MAIN,SUB) FORTLIB LIB(BIBLIO) PRINT(*) TERM
```

Так как по команде LOADGO происходит выполнение программы, то все наборы данных, используемые в программе, предварительно должны быть распределены постоянно или динамически.

5.2.5. Трансляция, редактирование и выполнение

В СРВ имеется возможность весь цикл обработки программы (трансляцию, редактирование и выполнение) произвести одной командой или подкомандой RUN. Команда RUN может использоваться в тех случаях, когда набор данных с текстом исходной программы уже создан и требуется получить только результаты счета. Подкоманда RUN команды EDIT позволяет выполнить полную обработку программы из набора данных, с которым работает команда EDIT. Форматы команды и подкоманды RUN:

```
RUN имя набора данных FORTSE
RUN
```

Первый операнд является позиционным, он определяет имя набора данных, содержащего исходную программу. Это имя записывается в таком же виде, как и в команде FORTSE (см. 5.2.1). Ключевой операнд FORTSE указывает, что для трансляции программы вызывается транслятор Фортран SE.

В подкоманде RUN операнды отсутствуют, так как имя и тип набора данных задаются в команде EDIT, к которой относится данная подкоманда.

Выполнение команды и подкоманды RUN равносильно действию команды FORTSE с режимами трансляции, установленными по умолчанию, и последующему выполнению команды LOADGO.

Используя команду (подкоманду) RUN, нельзя гибко управлять выводом результатов трансляции в требуемые наборы данных. Действующие по умолчанию режимы транслятора обеспечивают только вывод диагностических сообщений на АП и объектного модуля в набор данных. Имя набора данных с объектным модулем строится из имени набора данных с исходной программой. Тем не менее подкоманда RUN может ока-

заться полезной в тех случаях, когда программа практически отлажена и требуется выполнить счет по программе, не документируя результаты трансляции.

Пример. Иллюстрируются некоторые возможности команд СРВ для обработки программы транслятором Фортран SE. Средствами команды EDIT в набор данных USE.PROGM.FORT записывается программа, выполняющая вычисление площади треугольника по трем сторонам. Затем программа транслируется, редактируется и выполняется. Результаты трансляции (распечатка исходной программы и объектный модуль) не сохраняются.

Шаг 1. Ввод команды EDIT и установка режима ввода для записи программы в набор данных USER.PROGM.FORT. В команде EDIT запрашивается синтаксическая проверка предложений программы (операнд SCAN).

```
READY
edit progm new fortse scan
INPUT
```

Шаг 2. Запись исходной программы в набор данных.

```
00010        1 read(5,*)a,b,c
00020          p=0.5*(a+b+c)
00030          s=sqrt(p*(p—a)*(p—b)*(p—c))
00040          write(6,*) s
00050          stop
00060          end
00070          (пустая строка)
EDIT
save
SAVED
```

Шаг 3. Трансляция, редактирование и выполнение программы для получения результатов счета производятся по подкоманде RUN. По запросу программы (оператору с меткой 1) абонент вводит данные с АП.

```
run
SE COMPILER ENTERED
SOURCE ANALYZED
PROGRAM NAME=MAIN
* NO DIAGNOSTICS GENERATED
? 00001
2, 3, 4
  2.90473747
```

Шаг 4. Результаты вычислений совпали с ожидаемыми. Абонент завершает выполнение команды EDIT.

```
end
READY
```

5.2.6. Сеанс работы с транслятором Фортран SE

Рассмотрим пример сеанса работы, в котором иллюстрируются возможности обработки программ транслятором Фортран SE. В сеансе работы используется модифицированный вариант программы, рассмотренный в 5.1.2. Головной модуль вычисляет корни для нескольких квадратных уравнений. Коэффициенты уравнения вводятся по формату из набора данных, расположенного на дисках. В программе добавлена также подпрограмма OUTPUT, обеспечивающая вывод в набор данных коэффициентов и корней уравнения. Исходные тексты головного модуля и подпрограммы OUTPUT содержатся в последовательном наборе данных с именем PS.

Этап записи исходной программы исключен из примера, так как создание набора данных детально обсуждается в 5.1.2. Отлаженная подпрограмма SOLVE размещается в формате загрузочного модуля в разделе SOLVE библиотечного набора данных с именем BIBLIO. Текст подпрограммы приводится в 5.1.2. Сеанс состоит из 10 шагов.

Шаг 1. Определение начала и характеристик сеанса работы. Сеанс работы начинается вводом команды LOGON. Абонент указывает идентификатор TERM и имя FTEDIT процедуры LOGON. Характеристики сеанса работы он устанавливает такими же, как и в 5.1.2.

Шаг 2. Трансляция головного модуля и подпрограммы OUTPUT. Трансляция исходной программы, включающей головной модуль и подпрограмму OUTPUT, выполняется по команде FORTSE. Режимы трансляции задают вывод объектных модулей в набор данных с именем OBJ (операнд LOAD('OBJ') и вывод диагностических сообщений об ошибках на АП (режим TERM предполагается по умолчанию). Имена наборов данных указываются в команде в апострофах, поэтому идентифицирующий и описательный уточнители добавляться к именам не будут. При вводе команды FORTSE абонент допускает ошибку: вместо имени PS набирает имя PC. После просмотра системного каталога СРВ выдает сообщение с требованием ввести правильное имя. Абонент вводит имя PS.

Транслятор выдает сообщение SE COMPILER ENTERED о начале трансляции. Во время трансляции программы в головном модуле обнаружена ошибка: не определена метка объявления FORMAT с номером 2. На отсутствие этой метки указывают сообщения с идентификаторами IGI002I и IGI022I. Последующие сообщения информируют абонента о том, что анализ головного модуля, которому присвоено имя MAIN, закончен и что в модуле обнаружены две ошибки. Трансляция подпрограммы OUTPUT прошла успешно.

Шаг 3. Корректировка исходной программы. Чтобы выполнить корректировку исходной программы, абонент вводит команду EDIT. В команде EDIT он указывает имя набора данных PS, тип данных FORTSE и диспозицию набора данных OLD. Прежде чем скорректировать строку с номером 90, по подкоманде LIST абонент выводит текст программы на АП. Скорректированный набор данных сохраняется по подкоманде SAVE. По подкоманде END выполняется переход в режим ввода команд СРВ.

Шаг 4. Повторная трансляция программы. Транслируется скорректированная программа. Отсутствие диагностических сообщений означает, что трансляция прошла успешно. Объектные модули сохраняются в наборе данных с именем OBJ.

Шаг 5. Редактирование программы. Редактирование программы производится по команде LINK. В команде LINK указывается имя OBJ набора данных с объектными модулями (головным модулем и подпрограммой OUTPUT) и имя раздела SQUARE набора данных ZAGR, в который после редактирования помещается загрузочный модуль программы. При редактировании используется библиотека программ Фортрана (операнд FORTLIB) и личная библиотека BIBLIO, содержащая загрузочный модуль подпрограммы SOLVE.

Шаг 6. Распределение наборов данных рабочей программы для предварительного счета. Чтобы убедиться в том, что программа работает верно, абонент планирует выполнить предварительный счет, в котором данные вводит с АП, результаты вычисления также направляет на АП. По команде ALLOCATE абонент распределяет требуемые ему наборы данных на АП.

Шаг 7. Выполнение предварительного счета. Выполнение программы происходит по команде CALL. В команде CALL указывается имя набора данных с загрузочным модулем ZAGR(SQUARE). Во время выполнения программы на АП выдается запрос на ввод данных. В ответ на этот запрос абонент вводит коэффициенты уравнения по формату F6.2. Корни уравнения выводятся на АП в виде действительной и мнимой части. Так как результаты вычисления оказались верными, абонент решает прекратить выполнение программы. Для этого при очередном запросе на ввод он вызывает прерывание по сигналу ВНИМАНИЕ вводом символа %.

Шаг 8. Перераспределение наборов данных. После отладки абонент планирует выполнение программы для получения результатов счета, при котором коэффициенты уравнения вводятся из набора данных с именем DATA, а результаты вычислений направляются в системный выводной набор данных. Чтобы обеспечить такой режим работы, абонент вначале по команде FREE отменяет распределение используемых в программе наборов данных на АП, а затем заново распределяет их по команде ALLOCATE.

Шаг 9. Повторное выполнение программы. Вызов программы для вычисления значений корней уравнений, коэффициенты которых вводятся из набора данных, производится по команде CALL. Значения корней уравнений в отредактированном формате поступают в системный выводной набор данных. Последующая печать этого набора данных может быть выполнена в СРВ или в режиме пакетной обработки.

Шаг 10. Завершение сеанса работы. Сеанс работы завершается по команде LOGOFF.

Протокол сеанса работы имеет следующий вид:

```
logon term proc(ftedit)                                    Шаг 1
TERM LOGON IN PROGRESS AT 20:01:30 ON MARCH 3, 1981
NO BROADCAST MESSAGES
READY
terminal input(%)
READY
profile msgid prompt
READY
fortse 'pc' load('obj')                                    Шаг 2
IHM78107I DATA SET PC NOT IN CATALOG
IKJ56703A REENTER —
'ps'
SE COMPILER ENTERED
00000090     FORMAT (3F6.2)
                ↑
01) IGI002I LABEL
IGI022I UNDEFINED LABELS
             2
SOURCE ANALYZED
PROGRAM NAME = MAIN
*002 DIAGNOSTICS GENERATED, HIGHEST SEVERITY CODE
    IS 8
SOURCE ANALYZED
PROGRAM NAME = OUTPUT
*    NO DIAGNOSTICS GENERATED
      * STATISTICS * 002 DIAGNOSTICS THIS STEP␣:
READY
edit 'ps' old fortse                                       Шаг 3
EDIT
list
00010        COMMON X1REAL,X1IMAG,X2REAL,X2IMAG
00020        NPAGE = 1
00030 5      WRITE(2,3)NPAGE
00040 3      FORMAT(1H1,5X,1HA,9X,1HB,9X,1HC,6X,'X1REAL',4X,
00050       *'X1IMAG',4X,'X2REAL',4X,'X2IMAG',2X,'PAGE = ',I4//)
00060        NPAGE=NPAGE+1
00070        LINES=0
00080 1      READ(1,2,END=4)A,B,C
00090        FORMAT(3F6.2)
00100        CALL SOLVE(A,B,C)
00110        CALL OUTPUT(A,B,C)
00120        LINES=LINES+1
00130        IF(LINES—12)1,5,5
00140 4      STOP
00150        END
00160        SUBROUTINE OUTPUT(A,B,C)
00170        COMMON X1REAL,X1IMAG,X2REAL,X2IMAG
00180        IF(X1IMAG) 90,91,90
00190 91     WRITE(2,95)A,B,C,X1REAL,X2REAL
00200 95     FORMAT('0',4E10.2,10X,E10.2)
00210        RETURN
00220 90     IF(X1REAL) 100,101,100
00230 101    IF(X2REAL) 100,102,100
00240 102    WRITE(2,103)A,B,C,X1IMAG,X2IMAG
```

```
00250 103  FORMAT('0',3E10.2,10X,E10.2,10X,E10.2)
00260      RETURN
00270 100  WRITE(2,110)A,B,C,X1REAL,X1IMAG,X2REAL,X2IMAG
00280 110  FORMAT('0',7E10.2)
00290      RETURN
00300      END
IKJ52500I  END OF DATA
90    2    format(3f6.2)
save
SAVED
end
READY
fortse 'ps' load('obj')                                    Шаг 4
. . .
        *STATISTICS*  NO DIAGNOSTICS THIS STEP:
READY
link 'obj' load('zagr(square)') fortlib lib('biblio')      Шаг 5
READY
allocate fi(ft01f001) da(*)                                Шаг 6
READY
allocate fi(ft02f001) da(*)
READY
call 'zagr(square)'                                        Шаг 7
A     B      C     X1REAL X1IMAG X2REAL X2IMAG PAGE= 1
2.    30.    50.
0.20E+01 0.30E+02 0.50E+02  —0.19E+01      —0.13E+02
%
!
READY
free fi(ft01f001 ft02f001)                                 Шаг 8
READY
allocate fi(ft01f001) da('data') old
READY
allocate fi(ft02f001) sysout
READY
call 'zagr(square)'                                        Шаг 9
READY
logoff                                                     Шаг 10
IKJ56470I  TERM LOGGED OFF TSO AT 20:25:15 ON MARCH 3,
1981+
```

Глава 6

ОБРАБОТКА ПРОГРАММ НА ФОРТРАНЕ В ПАКЕТНОМ РЕЖИМЕ

Трансляторы Фортран SE и Фортран CC и программа преобразования могут выполняться не только в СРВ, но и в пакетном режиме.

6.1.1. Использование основной памяти и наборов данных

Минимальный объем основной памяти, необходимый для выполнения трансляторов Фортран SE и Фортран CC в пакетном режиме, равен соответственно 90 и 88 Кбайт. Эта память используется для размещения самого транслятора, таблиц и рабочих полей, которые строятся во время трансляции.

Вызов транслятора Фортран SE выполняется по оператору

```
//[имя шага] EXEC PGM=IGIFORT[,PARM='режим1,режим2,...']
```

Вызов транслятора Фортран CC выполняется по оператору

```
//[имя шага] EXEC PGM=IGKFTN[,PARM='режим1,режим2,...']
```

За один вызов каждого транслятора можно протранслировать несколько программных модулей. Наборы данных, используемые во время трансляции, приводятся в табл. 8.

В этой таблице также указываются стандартные групповые имена устройств, на которых располагаются наборы данных трансляторов, и входные классы, принятые в операционной системе и используемые в каталогизированных процедурах. Характеристики, устанавливаемые трансляторами по умолчанию для наборов данных, содержащих исходную программу в стандартном формате, приводятся в табл. 9. Длина блока каждого набора данных может быть изменена с помощью операнда DCB оператора DD.

Символы, определяющие формат записей, означают: A — записи содержат управляющие символы, B — записи объединены в блоки, F — записи имеют фиксированную длину, S — в наборе данных нет укороченных блоков или незаполненных дорожек, за исключением последнего блока или дорожки.

6.1.2. Режимы трансляции

Режимы трансляции предназначаются для указания формата исходной программы и управления выводом результатов трансляции. Они задаются в параметре

Таблица 8

Набор данных	Имя DD	Устройство ввода-вывода	Групповое имя, выходной класс
Входной набор данных, содержащий исходную программу	SYSIN	Перфокарточное устройство ввода	SYSSQ
		Накопитель на магнитной ленте	
		Накопитель на магнитных дисках	
Выходной набор данных, содержащий: а) распечатки исходной программы и сообщений; б) распечатки объектного модуля и распределения памяти (только для транслятора Фортран SE)	SYSPRINT	Печатающее устройство	SYSSQ, A
		Накопитель на магнитной ленте	
		Накопитель на магнитных дисках	
Выходной набор данных, содержащий объектный модуль	SYSPUNCH	Перфокарточное устройство вывода	SYSCP, B
		Накопитель на магнитной ленте	SYSSQ, B
		Накопитель на магнитных дисках	SYSSQ, B
Выходной набор данных, содержащий объектный модуль для Редактора или Загрузчика (только для транслятора Фортран SE)	SYSLIN	Перфокарточное устройство вывода	SYSCP
		Накопитель на магнитной ленте	SYSSQ
		Накопитель на магнитных дисках	SYSDA SYSSQ
Входные данные для загрузочного модуля (только для транслятора Фортран CC)	SYSGO	Перфокарточное устройство ввода	SYSSQ
		Накопитель на магнитной ленте	
		Накопитель на магнитных дисках	

Таблица 9

Имя DD	Длина записи LRECL	Формат записи RECFM		Длина блока BLKSIZE	Число буферов BUFNO
		Фортран CC	Фортран SE		
SYSIN	80	FB	FB	80	2
SYSPRINT	120	FA	FBSA	120	2
SYSPUNCH	80	FBS	FBSA	80	2
SYSLIN	80	—	FBS	80	2

PARM оператора EXEC. Если какой-либо режим пользователь не определяет, действует режим, установленный при генерации системы программирования Фортран. Режимы транслятора Фортран SE и транслятора Фортран CC приводятся в табл. 10.

Таблица 10

Режим		Действие
Фортран CC	Фортран SE	
DECK	DECK	Вывод объектного модуля в набор данных, определенный оператором DD с именем SYSPUNCH
NODECK	NODECK	Отмена режима DECK
—	ID	Указание на включение в объектный модуль номеров операторов, вызывающих внешние процедуры
—	NOID	Отмена режима ID
FIXED	—	Ввод исходной программы в стандартном формате
FREE	—	Ввод исходной программы в свободном формате
GO	—	Построение объектного модуля в основной памяти, редактирование и выполнение его сразу же после трансляции (при отсутствии ошибок с кодом выше 4)
NOGO	—	Отмена режима GO
LINECNT=n	LINECNT=n	Задание количества строк (n) на листе распечатки
—	LIST	Вывод распечатки объектного модуля в набор данных, определенный оператором DD с именем SYSPRINT
—	NOLIST	Отмена режима LIST
LMSG	—	Вывод сообщений об ошибках в расширенном формате в набор данных, определенный оператором DD с именем SYSPRINT

Продолжение табл. 10

Режим		Действие
Фортран CC	Фортран SE	
SMSG	—	Вывод сообщений об ошибках в сжатом формате в набор данных, определенный оператором DD с именем SYSPRINT
—	LOAD	Вывод объектного модуля в набор данных, определенный оператором DD с именем SYSLIN, для последующего использования Редактором или Загрузчиком
—	NOLOAD	Отмена режима LOAD
	NAME	Определение имени головного модуля. Это имя идентифицирует выводимые распечатки и перфокарты
—	MAP	Вывод распечатки распределения памяти для элементов исходной программы в набор данных, определенный оператором DD с именем SYSPRINT
—	NOMAP	Отмена режима MAP
SOURCE	SOURCE	Вывод распечатки исходной программы в набор данных, определенный оператором DD с именем SYSPRINT
NOSOURCE	NOSOURCE	Отмена режима SOURCE
TEST	TEST	Вывод объектного модуля для выполнения отладки с помощью Диалогового отладчика

Пример.

```
//STEP EXEC PGM=IGKFTN,PARM='FIXED,LMSG,SOURCE,GO'
```

Этот оператор вызывает транслятор Фортран CC. Исходная программа подготовлена в стандартном формате. В результате трансляции будет выдана распечатка текста исходной программы вместе с сообщениями об ошибках в расширенном формате. Для синтаксически правильной программы транслятор построит и выполнит рабочую программу. Для программы с ошибками режим GO игнорируется.

6.1.3. Входная информация трансляторов

Входной информацией трансляторов являются исходные программы на языке Фортран. Исходная програм-

ма для транслятора Фортран SE готовится в стандартном формате, а для транслятора Фортран CC — в стандартном или в свободном формате. Стандартный формат исходной программы указывается режимом FIXED, свободный формат — режимом FREE. Правила записи программ в стандартном и свободном форматах описаны в 3.2.3. Исходная программа, подготовленная в свободном формате для транслятора Фортран CC, может быть обработана транслятором Фортран SE, если ее предварительно преобразовать в стандартный формат с помощью программы преобразования CONVERT.

6.1.4. Преобразование формата предложений исходной программы

Преобразование предложений исходной программы из свободного формата в стандартный и наоборот выполняет программа CONVERT. Программа преобразования использует 3 набора данных (табл. 11).

Входной набор данных может содержать несколько программных модулей и состоять из записей фиксированной или переменной длины. Если предложения подготовлены в свободном формате, набор данных состоит из записей фиксированной длины по 80 байт или записей переменной длины, не превосходящих 1326 байт. Если предложения подготовлены в стандартном формате, набор данных состоит из записей фиксированной длины по 80 байт.

Выходной набор данных с предложениями Фортрана, преобразованными в заданный формат, также может состоять из записей фиксированной или переменной длины. Если производится преобразование предложений из свободного формата в стандартный, выходной набор данных будет состоять из записей фиксированной длины по 80 байт. При создании выходного набора данных устанавливаются следующие характеристики записей: RECFM=FB, BLKSIZE=80, LRECL=80. Если производится преобразование из стандартного формата в свободный, набор данных будет состоять из записей переменной длины. При создании набора данных устанавливаются следующие характеристики записей: RECFM=VB, BLKSIZE=89, LRECL=85.

Таблица 11

Набор данных	Имя DD	Устройство ввода-вывода	Групповое имя, выходной класс
Входной набор данных, содержащий исходную программу в стандартном или свободном формате	SYSUT1	Перфокарточное устройство ввода	SYSSQ
		Накопитель на магнитной ленте	
		Накопитель на магнитных дисках	
Выходной набор данных, содержащий исходную программу, преобразованную в заданный формат	SYSUT2	Перфокарточное устройство вывода	SYSCP, B
		Накопитель на магнитной ленте	SYSSQ, B
		Накопитель на магнитных дисках	SYSDA, SYSSQ, B
Выходной набор данных, содержащий распечатку исходной программы, преобразованной в заданный формат, и сообщения об ошибках	CNVOUT	Печатающее устройство	SYSSQ, A
		Накопитель на магнитной ленте	
		Накопитель на магнитных дисках	

Вызов программы преобразования производится по оператору

//[имя шага]EXEC PGM=IGKCNV[,PARM='режим1,режим2,...']

Режимы работы программы преобразования указываются в параметре PARM оператора EXEC. Если какой-либо из режимов опущен, действует режим, установленный программой преобразования по умолчанию. В параметре PARM могут быть указаны режимы, приведенные ниже (значения, устанавливаемые по умолчанию, подчеркнуты).

FIXED — указание на преобразование предложений из свободного формата в стандартный.

FREE — указание на преобразование предложений из стандартного формата в свободный.

LIST — вывод распечатки преобразованных предложений в набор данных, определенный оператором DD с именем CNVOUT.

NOLIST — отмена режима LIST.

SEQ — перенумерация строк исходной программы с шагом 10 при преобразовании предложений. Первой строке присваивается номер 10. В случае режима FIXED номер помещается в позиции 73—80, а в случае режима FREE — в позиции 1—8 каждой строки.

NOSEQ — отмена режима SEQ. Номера строк исходной программы не изменяются и переносятся соответственно в позиции 73—80 (режим FIXED) или 1—8 (режим FREE) каждой строки.

LINECNT=n — задание количества строк (n) на листе распечатки.

При выполнении программы преобразования могут быть выданы сообщения об ошибках, которые имеют следующий формат:

```
IGK xxxI текст
```

где xxx — номер сообщения, текст — текст сообщения.

Для каждого сообщения устанавливается код серьезности ошибки. Обработка продолжается, если код серьезности равен нулю, и прекращается, если код серьезности равен 4. Сообщения, выдаваемые программой преобразования, приводятся в приложении 1.

Пример. Исходная программа, подготовленная на перфокартах в свободном формате, должна быть преобразована в стандартный формат и помещена в набор данных с именем FORTIN. Задание на выполнение преобразования имеет вид:

```
//CONVERT1 JOB (561,7808),'ПЕТРОВ',MSGLEVEL=(1,1)
//         EXEC PGM=IGKCNV
//CNVOUT   DD   SYSOUT=A
//SYSUT2   DD   DSN=FORTIN,UNIT=SYSSQ,
//              DISP=(NEW,CATLG),SPACE=(80,(200,100))
//SYSUT1   DD   *
Исходная программа в свободном формате
/*
//
```

6.1.5. Выходная информация трансляторов

Выходная информация трансляторов определяется режимами, установленными при включении трансляторов в операционную систему либо заданными пользователем в операторе EXEC. В соответствии с действующими режимами для каждой транслируемой исходной программы выдаются: распечатка исходной программы, распечатка распределения памяти для элементов исходной программы (только транслятором Форт-

ран SE), распечатка объектного модуля (только транслятором Фортран SE), объектный модуль в наборе данных. В случае обнаружения ошибок в исходной программе трансляторы выдают диагностические сообщения.

Каждый лист распечаток, выдаваемых трансляторами, начинается стандартной строкой, которая идентифицирует соответствующий транслятор и содержит имя транслируемого программного модуля (выдается только транслятором Фортран SE), дату, время начала трансляции и номер листа. Транслятор Фортран SE присваивает головному модулю имя MAIN или имя, заданное в операнде PARM. Модулю-процедуре всегда присваивается имя из заголовка процедуры.

Распечатка исходной программы. При задании режима SOURCE трансляторы Фортран SE и Фортран CC выводят распечатку исходной программы в набор данных, определенный оператором DD с именем SYSPRINT. Каждому предложению исходной программы присваивается порядковый номер. Транслятор Фортран SE ведет нумерацию относительно начала каждого программного модуля независимо от количества модулей, транслируемых в одном шаге. Транслятор Фортран CC ведет сквозную нумерацию предложений всех модулей, обрабатываемых в одном шаге.

Транслятор Фортран CC при указании режима GO вызывает Загрузчик. В этом случае после распечатки исходной программы выводится таблица, содержащая схему распределения памяти рабочей программы.

Распечатка распределения памяти. При задании режима MAP транслятор Фортран SE выводит распечатку распределения памяти в набор данных, определенный оператором DD с именем SYSPRINT. В распечатке содержатся сведения о группах элементов исходной программы, аналогичных тем, которые выдаются транслятором Фортран ST [1].

Распечатка объектного модуля. При задании режима LIST транслятор Фортран SE выводит в набор данных, определенный оператором DD с именем SYSPRINT, распечатку объектного модуля в формате команд языка Ассемблера. На рис. 1 приведен вид распечатки объектного модуля. Графы распечатки содержат информацию, аналогичную той, которая выдается транслятором Фортран ST [1].

OC EC FORTRAN SE 85—06.80 MAIN DATE=80123 11/10/23 PAGE 0003

LOCATION	STA NUM	LABEL	OP	OPERAND		BCD OPERAND
000000			BC	15,12(0,15)		
000004			DC	06D4C1C9		(имя объектного моду-
000008			DC	D5404040	З	ля)
00000C			STM	14,12,12(13)	А	
000010			LM	2,3,40(15)	Г	
000014			LR	4,13	О	
000016			L	13,36(0,15)	Л	
00001A			ST	13,8(0,4)	О	
00001E			STM	3,4,0(13)	В	
000022			BCR	15,2	О	
000024			DC	00000000	К	A4 (адр. обл. сохр.)
000028			DC	00000000		A20 (адрес пролога)
00002C			DC	00000000		A36 (адрес эпилога)
000184		A36	L	13,4(0,13)		
000188			L	14,12(0,13)		
00018C			LM	2,12,28(13)		
0001C0			MVI	12(13),255		
0001C4			BCR	15,14		
0001C6		A20	L	15,140(0,13)		IBCOM#
0001CA			LR	12,13		
0001CC			LR	13,4		
0001CE			BAL	14,64(0,15)		
0001D2			LR	13,12		
0001D4	2		L	0,376(0,13)		
0001D8			ST	0,200(0,13)		NPAGE
0001DC	3	5	L	15,140(0,13)		IBCOM#
0001E0			BAL	14,4(0,15)		
0001E4			DC	00000006		
0001E8			DC	00000118		

0001EC			BAL	14,8(0,15)	
0001F0			DC	0450D0CB	
0001F4			BAL	14,16(0,15)	
0001F8	5		L	0,200(0,13)	NPAGE
0001FC			A	0,376(0,13)	
000200			ST	0,200(0,13)	NPAGE
000204	6		L	0,320(0,13)	
000208			ST	0,204(0,13)	LINES
00020C	7	14	L	15,140(0,13)	IBCOM#
			BAL	14,0(0,15)	
	. . . . .	. . . . .	. . . . .	. . . . .	. . . . .
0002C2			LTR	0,0	
0002C4			L	14,104(0,13)	14
0002CB			BCR	4,14	
0002CA			L	14,100(0,13)	5
0002CE			END		

Рис. 1. Распечатка объектного модуля, выдаваемая транслятором Фортран SE

Распечатка сообщений. Сообщения, выдаваемые трансляторами, можно разделить на информационные сообщения (только для транслятора Фортран SE) и диагностические сообщения об ошибках в исходной программе и о состоянии трансляции. Информационные сообщения выводятся в конце всех распечаток для каждого программного модуля и содержат сведения о действующих режимах, количестве предложений в программном модуле, размере построенного объектного модуля, количестве обнаруженных ошибок и наивысший код серьезности ошибок. Если в одном шаге транслировалось несколько программных модулей, то дополнительно транслятор Фортран SE выдает сообщение об общем количестве ошибок, обнаруженных во всех программных модулях.

Для каждой ошибки трансляторы устанавливают код серьезности. Наивысший код серьезности ошибок помещается в регистр 15 в качестве кода возврата, передаваемого трансляторами управляющей программе. Код серьезности принимает значение 0, 4, 8 или 16 и означает следующее:

0 — ошибка не грубая. Результаты выполнения рабочей программы могут быть недостоверными;

4 — сообщение предупреждающее. Рабочая программа будет выполняться, но результаты будут недостоверными;

8 — ошибка грубая. Транслятор Фортран CC заканчивает текущую фазу, и трансляция прекращается. Транслятор Фортран SE продолжает обработку;

16 — ошибка грубая. Трансляция прекращается.

Транслятор Фортран SE выдает диагностические сообщения об ошибках в программе непосредственно за предложениями, в которых обнаружены ошибки, и в конце распечатки. Позиция предложения, в которой допущена ошибка, отмечается символом ↑ в следующей за предложением строке. Сообщения для одного предложения могут занимать одну или несколько строк. В каждой строке может содержаться несколько сообщений. Сообщение имеет следующий формат:

```
nn) IGIxxxI текст
```

где nn — порядковый номер сообщения в строке предложения; xxx — номер сообщения; текст — текст сообщения. Сообщения транслятора Фортран SE в основ-

ном идентичны сообщениям транслятора Фортран ST [1].

Транслятор Фортран CC выводит сообщения после распечатки всех программных модулей. Каждое сообщение содержит номер предложения, в котором была допущена ошибка. Если указан режим SMSG, сообщения об ошибках выдаются в краткой форме. При указании режима LMSG выдается подробный текст сообщений. Сообщение имеет следующий формат:

```
IGKxxxI yyyyyyyy текст
```

где xxx — номер сообщения, yyyyyyyy — номер ошибочного предложения, текст — текст сообщения.

Диагностические сообщения о состоянии трансляции (например, о недостаточном количестве памяти для таблиц, об ошибках ввода-вывода) могут появиться в любом месте распечаток, выдаваемых трансляторами.

Объектный модуль в наборе данных. При задании режима LOAD (для транслятора Фортран SE) объектный модуль выводится в набор данных, определенный оператором DD с именем SYSLIN, а при задании режима DECK (для трансляторов Фортран SE и Фортран CC) объектный модуль выводится в набор данных, определенный оператором DD с именем SYSPUNCH. Каждая запись объектного модуля представляется в формате перфокарт. Объектный модуль состоит из перфокарт четырех типов: ESD, TXT, RLD и END. В первой колонке каждой перфокарты содержится комбинация пробивок 12—2—9, указывающая на принадлежность перфокарты объектному модулю. В колонках 2—4 содержится тип перфокарты, определяемый соответственно символами ESD, TXT, RLD или END. Колонки 73—80 используются для идентификации перфокарты, в них размещаются первые четыре символа имени исходного модуля и последовательный номер перфокарты в колоде.

Размещение объектного модуля в основной памяти. В тех случаях, когда пользователь вынужден прибегать к анализу объектного кода, построенного транслятором, ему потребуется знать расположение элементов исходной программы в основной памяти. На рис. 2 представлен порядок размещения частей объектного модуля в памяти.

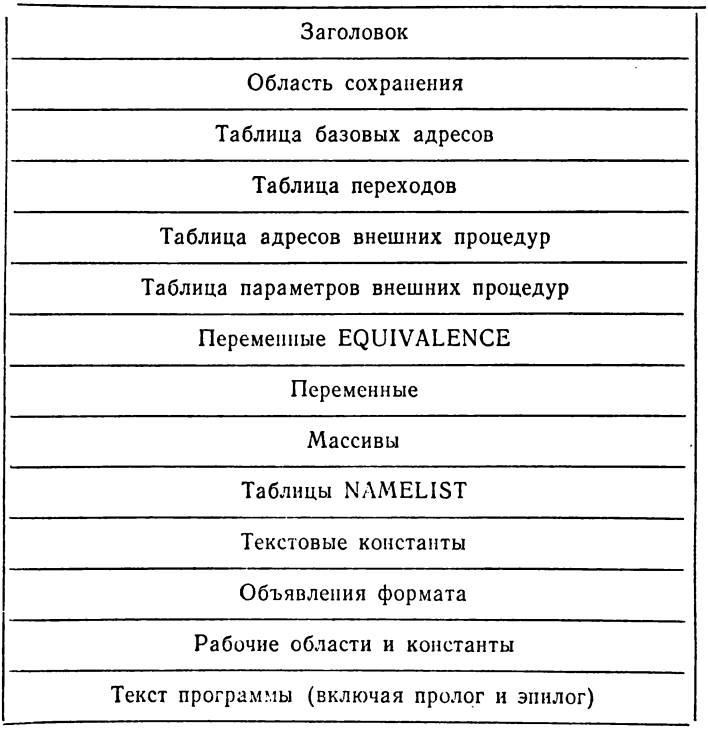


Рис. 2. Размещение объектного модуля в основной памяти

Ниже описывается использование общих регистров в объектном модуле.

0 — для передачи значения функции вызывающей программе и накопления промежуточных результатов.

1 — для адреса списка параметров, передаваемого вызывающей программой, и накопления промежуточных результатов.

2, 3 — для накопления промежуточных результатов.

4—9 — для хранения промежуточных результатов.

10—12 — для значений, загружаемых из таблицы базовых адресов.

13 — для хранения адреса области сохранения объектного модуля и доступа к таблице базовых адресов.

14 — для адреса возврата в вызывающую программу и адреса команды перехода во время выполнения программы.

15 — для адреса входа в процедуру при обращении к ней.

Независимо от текста исходной программы объектный модуль содержит заголовок, пролог и эпилог. Выполнение объектного модуля начинается с заголовка. Заголовок запоминает некоторую информацию в обла-

стях сохранения вызывающей и вызываемой программ и передает управление прологу. Пролог выполняет действия по инициализации программного модуля и передает управление командам, построенным по операторам исходной программы. Эпилог заканчивает выполнение объектного модуля и организует возврат в вызывающую программу.

При обработке программ транслятором Фортран SE некоторые части объектного кода можно видеть на распечатке. По режиму LIST транслятор Фортран SE выдает распечатку объектного модуля, которая содержит заголовок, пролог, эпилог и объектный код операторов исходной программы. Остальные части объектного модуля, указанные на рис. 2, на распечатку не выдаются.

Заголовок занимает первые 48 байт объектного модуля (рис. 1) и включает команды, выполняющие следующие действия:

запоминание содержимого общих регистров 14, 15, 0—12 вызывающей программы в ее области сохранения, начиная с четвертого слова;

занесение в регистр 2 адреса пролога основного или дополнительного входа, в регистр 3 — адреса эпилога основного или дополнительного входа, в регистр 4 — адреса области сохранения вызывающей программы, в регистр 13 — адреса области сохранения данной программы;

запоминание адреса области сохранения данной программы в третьем слове области сохранения вызывающей программы, адреса эпилога и адреса области сохранения вызывающей программы в первом и втором словах области сохранения данной программы;

передачу управления соответствующему прологу программы.

В модуле-процедуре заголовок строится для основного входа и каждого дополнительного входа. Текст заголовка головного модуля и основного входа в модуль-процедуру (SUBROUTINE или FUNCTION) приведен на рис. 1. Для каждого дополнительного входа в модуле-процедуре строится заголовок вида:

```
BC      15,12(0,15)
DC      AL1 (длина имени входа)
DC      Cln (имя входа)
STM     14,12,12(13)
LM      2,3,32(15)
```

```
L       15,28(0,15)
BC      15,20(0,15)
DC      (адрес основного входа в модуль-процедуру)
DC      (адрес пролога для данного входа)
DC      (адрес эпилога для данного входа)
```

В модуле-процедуре заголовки дополнительных входов располагаются после заголовка основного входа.

Область сохранения занимает 72 байта и предназначается для сохранения информации, необходимой для связи между вызывающей и вызываемой программами. Подробную информацию о структуре этой области можно найти в [1]. Относительный адрес области сохранения для головного модуля и модуля-процедуры, имеющего только основной вход, равен 48.

Таблица базовых адресов содержит относительные адреса, используемые как базовые при обращении к данным. Разность между двумя смежными адресами в таблице равна или превышает 4096 байт. Для массива, размер которого больше 4096 байт, в таблице базовых адресов строится одна запись. Все адреса в таблице являются относительными.

Таблица переходов содержит относительные адреса помеченных операторов, на которые имеются ссылки в операторах управления, а также начальные адреса объектного кода, построенного для внутренних функций. Кроме того, в таблицу включаются адреса операторов, помечаемых транслятором (например, операторы начала цикла DO, операторы, следующие за логическими операторами IF). Перед выполнением перехода адрес из таблицы переходов загружается в регистр 14.

Таблица адресов внешних процедур содержит по одной записи для каждой внешней процедуры, на которую имеется ссылка в исходном модуле. Размер таблицы зависит от количества процедур, используемых в исходной программе. Перед обращением к внешней процедуре адрес из таблицы внешних процедур загружается в регистр 15.

Таблица параметров внешних процедур содержит адреса параметров для всех вызываемых модулей-процедур. Перед вызовом модуля-процедуры адрес соответствующего участка таблицы параметров пересылается в общий регистр 1. С помощью этого адреса формируется обращение к параметрам внешней процедуры. Нулевой бит слова, содержащего адрес последнего параметра внешней процедуры, установлен в 1.

Переменные EQUIVALENCE. Эта область памяти содержит переменные и массивы, перечисленные в списках EQUIVALENCE и не описанные в предложениях COMMON.

Переменные. В этой области объектного модуля размещаются все переменные, не описанные в предложениях COMMON и не являющиеся элементами списков EQUIVALENCE.

Массивы. В этой области объектного модуля размещаются все массивы, не описанные в предложениях COMMON и не являющиеся элементами списков EQUIVALENCE.

Таблица NAMELIST строится для каждого списка, имя которого указано в объявлении списка. Таблица содержит одну запись для каждого символического имени, которое указано в списке. Заголовок таблицы имеет длину 8 байт и содержит имя списка.

Текстовые константы. Эта область содержит текстовые константы, используемые в качестве фактических параметров при обращении к внешним процедурам.

Объявления формата включают информацию из предложений FORMAT, расположенную в порядке их появления в исходной программе. Информация содержит все спецификации, кроме слова FORMAT.

Рабочие области и константы. Эта область всегда начинается на границе двойного слова. Она содержит константы необязательно в том порядке, в каком они указаны в исходной программе, а также рабочие области для временного размещения промежуточных результатов вычислений. Константы исходной программы, которые могут быть размещены в машинных командах как непосредственные данные, не включаются в эту область. В этой области находятся также константы, которые создаются транслятором.

Текст программы содержит один или несколько эпилогов и прологов программы, объектные коды для внутренних функций и объектные коды для операторов. Коду для внутренней функции предшествует команда перехода к первому выполняемому оператору программы.

Пролог головного модуля выполняет подготовительные действия для всей рабочей программы путем обращения к программе библиотеки Фортрана с точкой входа IBCOM#. Действия по инициализации заключаются в следующем:

макрокомандой SPIE запрашивается обработка программных прерываний средствами библиотеки Фортрана;

макрокомандой STAE задается обработка аварийного окончания рабочей программы средствами библиотеки Фортрана;

открывается набор данных для вывода сообщений об ошибках в рабочей программе. В стандартном случае этот набор данных имеет номер 6.

Для внешней процедуры пролог строится для каждого ее входа. Пролог обеспечивает пересылку фактических параметров в память, отведенную для формальных параметров, и передачу управления на требуемый вход, реализующий алгоритм вычисления.

После выполнения вычислений по операторам исходной программы и действий по оператору RETURN управление в модуле-процедуре получает эпилог. Код для эпилога строится для каждого входа в модуль-процедуру. Он возвращает параметры в вызывающую программу, восстанавливает общие регистры 2—12 и 14 вызывающей программы. Признак успешного завершения процедуры X'FF' пересылается в нулевой байт четвертого слова области сохранения вызывающей программы. После выполнения указанных действий эпилог организует возврат в вызывающую программу. Эпилог для головного модуля управление не получает. Оператор STOP или END передает управление программам библиотеки Фортрана, которые организуют возврат в вызывающую программу.

6.2. ВЫПОЛНЕНИЕ

Протранслированная программа может быть выполнена в пакетном режиме операционной системы ОС только после обработки Редактором или Загрузчиком. Если обработка исходной программы производится транслятором Фортран CC в режиме GO, то транслятор вызывает Загрузчик для формирования и выполнения загрузочного модуля. Если трансляция исходной программы производится транслятором Фортран SE или транслятором Фортран CC в режиме NOGO, то пользователь самостоятельно должен планировать вызов Редактора или Загрузчика. Программа, приведенная Редактором к виду загрузочного модуля, выполняется по оператору

EXEC. В случае использования Загрузчика выполнение программы производится автоматически после ее редактирования.

Входной информацией рабочей программы являются данные, подготовленные на любом допустимом носителе данных для ввода по операторам Фортрана.

Выходную информацию рабочей программы составляют данные, выводимые в наборы данных операторами Фортрана на допустимые устройства вывода. Соответствие между номерами наборов данных, которые используются в операторах ввода-вывода, и именами операторов DD, а также сведения о типе и организации наборов данных приводятся в табл. 12. Наборы данных с номерами 5, 6, 7 используются соответственно как системные устройства ввода, печати и перфорации. Наборы данных с номерами 1—4, 8—99 могут использоваться как для ввода, так и для вывода и иметь последовательную, прямую и библиотечную организацию.

Во время выполнения рабочей программы могут выдаваться сообщения об ошибках и сообщения оператору.

Сообщения об ошибках выдаются программами библиотеки Фортрана в набор данных, определенный оператором DD с именем FT06F001. При включении библиотеки в операционную систему ОС можно задать один из двух способов обработки ошибок. При первом способе обработки после возникновения ошибочной ситуации выдается сообщение об ошибке, и выполнение программы прекращается. Исключение составляют ошибки, связанные с программными прерываниями, — в этом случае предпринимаются некоторые действия по корректировке результата, и выполнение рабочей программы продолжается. При втором способе обработки, называемом расширенной обработкой ошибок, после возникновения ошибочной ситуации выдается сообщение об ошибке, производится корректирующее действие, и выполнение программы продолжается. Корректирующие действия можно задавать стандартные, предусмотренные программами библиотеки Фортрана, и нестандартные, определенные пользователем. Подробная информация об использовании средств расширенной обработки ошибок приводится в [1].

Сообщения об ошибках имеют идентификатор IHOxxxI, где xxx — номер сообщения. За идентификато-

Таблица 12

Номер набора данных	Тип набора данных. Организация	Имя DD	Устройство ввода-вывода	Групповое имя, выходной класс
5	Вводной. Последовательная организация	FT05F001	Перфокарточное устройство ввода, накопитель на магнитной ленте, накопитель на магнитных дисках	SYSSQ
6	Выводной. Последовательная организация	FT06F001	Печатающее устройство, накопитель на магнитной ленте, накопитель на магнитных дисках	SYSSQ, A
7	Выводной. Последовательная организация	FT07F001	Перфокарточное устройство вывода, накопитель на магнитной ленте, накопитель на магнитных дисках	SYSSQ, SYSCP, B
1—4 8—99	Вводной и (или) выводной. Последовательная, прямая или библиотечная организация	FTxxFyyy (xx — номер набора данных, yyy — порядковый номер)	Перфокарточное устройство ввода, перфокарточное устройство вывода, печатающее устройство, накопитель на магнитной ленте, накопитель на магнитных дисках	SYSSQ, SYSDA

ром обычно следует текст сообщения. Для ошибок, приводящих к прекращению выполнения рабочей программы, сообщения сопровождаются распечаткой списка вызываемых процедур. Список процедур выводится в порядке, обратном порядку обращения к ним. Если используется средство расширенной обработки ошибок, то список процедур печатается всегда. Сообщения об ошибках, возможные при выполнении рабочей программы, в основном соответствуют сообщениям, выдаваемым ра-

бочей программой, обработанной транслятором Фортран ST [1].

Сообщения оператору выводятся при выполнении операторов PAUSE и STOP. Вывод производится на устройство связи с оператором.

6.3. КАТАЛОГИЗИРОВАННЫЕ ПРОЦЕДУРЫ

Пользователям Фортрана поставляется 13 каталогизированных процедур: 6 процедур для вызова транслятора Фортран SE и 7 процедур для вызова транслятора Фортран CC. Если эти процедуры не отвечают требованиям пользователя, он может создать свои собственные и поместить их в библиотеку процедур.

Для вызова транслятора Фортран SE предназначаются следующие каталогизированные процедуры: FTSEC (трансляция); FTSECL (трансляция и редактирование); FTSECLG (трансляция, редактирование и выполнение в трех шагах задания); FTSECG (трансляция, редактирование и выполнение в двух шагах задания); FTSELG (редактирование и выполнение в двух шагах задания); FTSEG (редактирование и выполнение в одном шаге задания).

В указанных процедурах редактирование программы производится Редактором или Загрузчиком. Если для редактирования используется Редактор, то для выполнения программы требуется самостоятельный шаг задания. В случае использования Загрузчика редактирование и выполнение программы производятся в одном шаге задания.

Для вызова транслятора Фортран CC предназначаются следующие каталогизированные процедуры: FTCCC (трансляция); FTCCCL (трансляция и редактирование); FTCCCLG (трансляция, редактирование и выполнение в трех шагах задания); FTCCCG (трансляция, редактирование и выполнение в двух шагах задания); FTCCCGO (трансляция, редактирование и выполнение в одном шаге задания); FTCCLG (редактирование и выполнение в двух шагах задания); FTCCG (редактирование и выполнение в одном шаге задания).

Пример 1. Трансляция с выводом распечатки исходной программы.

```
//PRIMER1   JOB
//          EXEC { FTSEC }
                 { FTCCC }
//FORT.SYSIN DD *
```

```
        Программные модули
/*
//
```

Пример 2. Трансляция с выводом распечатки исходной программы и с сохранением объектных модулей в наборе данных OBJMOD.

```
//PRIMER2    JOB
//          EXEC { FTSEC } ,PARM=DECK
                 { FTCCC }
//FORT.SYSPUNCH DD DSNAME=OBJMOD,DISP=(NEW,
//          CATLG),
//          UNIT=SYSDA,VOL=SER=FORTSE,
//          SPACE=(80,(200,100)),DCB=BLKSIZE=80
//FORT.SYSIN DD *
        Программные модули
/*
//
```

Пример 3. Трансляция и редактирование с сохранением загрузочного модуля в разделе LDMOD библиотечного набора данных LIB.

```
//PRIMER3    JOB
//          EXEC { FTSECL }
                 { FTCCCL }
//FORT.SYSIN DD *
        Программные модули
/*
//LKED.SYSLMOD DD DSNAME=LIB(LDMOD),UNIT=SYSDA,
//             VOL=SER=FORTSE,DISP=(NEW,CATLG),
//             SPACE=(1024,(20,10,1),RLSE),
//             DCB=BLKSIZE=1024
//
```

Пример 4. Трансляция, редактирование и выполнение в трех шагах задания.

```
//PRIMER4    JOB
//          EXEC{ FTSECLG }
               { FTCCCLG }
//FORT.SYSIN DD *
        Программные модули
/*
//GO. SYSIN  DD *
        Входные данные рабочей программы
/*
//
```

Пример 5. Трансляция, редактирование и выполнение в двух шагах задания.

```
//PRIMER5    JOB
//          EXEC{ FTSECG }
               { FTCCCG }
//FORT.SYSIN DD *
```

```
        Программные модули
/*
//GO. SYSIN  DD *
        Входные данные рабочей программы
/*
//
```

Пример 6. Трансляция, редактирование и выполнение в одном шаге задания (только для транслятора Фортран СС).

```
//PRIMER6     JOB
//           EXEC FTCCCGO
//FORT.SYSIN  DD *
        Программные модули
/*
//FORT.SYSGO DD *
        Входные данные рабочей программы
/*
//
```

Пример 7. Редактирование и выполнение в двух шагах задания. Объектный модуль хранится в каталогизированном наборе данных OBJMOD.

```
//PRIMER7      JOB
//             EXEC { FTSELG }
                  { FTCCLG }
//LKED.SYSLIN DD DSNAME=OBJMOD,DISP=OLD
//GO.SYSIN     DD *
        Входные данные рабочей программы
/*
//
```

Пример 8. Редактирование и выполнение в одном шаге задания. Объектный модуль хранится в каталогизированном наборе данных OBJMOD.

```
//PRIMER8   JOB
//          EXEC { FTSEG }
                 { FTCCG }
//GO.SYSLIN DD DSNAME=OBJMOD,DISP=OLD
//GO.SYSIN  DD *
        Входные данные рабочей программы
/*
//
```

Пример 9. Выполнение в одном шаге задания. Загрузочный модуль хранится в разделе LDMOD каталогизированного библиотечного набора LIB.

```
//PRIMER9 JOB
//        EXEC PGM=LDMOD
//STEPLIB DD   DSN=LIB,DISP=SHR
//FT06F001 DD  SYSOUT=A
//FT05F001 DD  *
        Входные данные рабочей программы
/*
//
```

Глава 7

ДИАЛОГОВАЯ ОТЛАДКА ПРОГРАММ

Диалоговая отладка предоставляет абоненту средства общения с рабочей программой во время ее выполнения. Инструментом этого общения является команда СРВ TESTFORT и ее подкоманды. Диалоговый отладчик, обрабатывающий команду TESTFORT и ее подкоманды, позволяет при задании отладочных действий использовать такие элементы исходной программы, как метки и номера строк операторов, символические имена переменных и массивов.

7.1. ПОДГОТОВКА ПРОГРАММЫ К ОТЛАДКЕ

Диалоговая отладка не предъявляет особых требований к исходной программе. Необходимо только, чтобы программные модули не содержали предложений отладки DEBUG, AT, TRACE ON, TRACE OFF и DISPLAY и чтобы строки предложений программных модулей были пронумерованы. Трансляция исходной программы, для которой планируется диалоговая отладка, выполняется с режимом TEST. По режиму TEST создается объектный модуль, в котором устанавливается связь с Диалоговым отладчиком и строятся дополнительные таблицы. Эти таблицы позволяют во время отладки программы обращаться к операторам Фортрана по меткам или номерам строк, а к переменным и массивам — по их символическим именам. Если исходная программа состоит из нескольких модулей, то достаточно протранслировать с режимом TEST только те модули, которые будут отлаживаться в диалоговом режиме. Головной модуль при трансляции должен получить имя MAIN.

В СРВ трансляция исходной программы производится по командам FORTCC и FORTSE. При использовании транслятора Фортран СС полученные объектные модули сохраняются в наборе данных, имя которого строится из имени набора данных с исходной программой в соответствии с правилами, изложенными в 5.1. Например, если исходная программа состоит из модулей MAIN, SOLVE и OUTPUT, которые содержатся в последовательном наборе данных IVANOV.SQUARE.FORT, то по команде FORTCC SQUARE TEST, введенной абонентом, работающим с идентификатором IVA-

NOV, объектные модули программы поместятся в последовательный набор данных IVANOV.SQUARE.OBJ. В отличие от транслятора Фортран CC транслятор Фортран SE позволяет сохранить объектные модули в наборе данных, имя которого задается в операнде LOAD команды FORTSE. Так, если объектные модули после трансляции необходимо сохранить в разделе SQRT библиотечного набора данных IVANOV.SORTAB.OBJ, абонент может ввести команду

FORTSE SQUARE TEST LOAD(SORTAB(SQRT))

Объектные модули для диалоговой отладки могут быть получены и в пакетном режиме.

7.2. ВЫЗОВ ДИАЛОГОВОГО ОТЛАДЧИКА

Команда TESTFORT, вызывающая Диалоговый отладчик, имеет формат:

TESTFORT (список имен наборов данных)
[LIB(список имен наборов данных)]
[SOURCE [(список имен наборов данных)]]
[NOSOURCE]
[PRINT [(имя набора данных)]] [RES]
[NOPRINT] [NORES]

Команда TESTFORT содержит один позиционный и четыре ключевых операнда. Позиционный операнд указывает имена наборов данных с объектными и загрузочными модулями отлаживаемой программы. Отлаживаемые модули обычно представляются в формате объектных модулей. Если набор данных последовательный и его имя составлено в соответствии с действующими в СРВ правилами именования, то в команде TESTFORT указывается только имя набора данных, присваиваемое абонентом. Например, если объектные модули отлаживаемой программы содержатся в последовательном наборе данных IVANOV.SQUARE.OBJ, то абонент, работающий с идентификатором IVANOV, может ввести команду TESTFORT SQUARE.

Для библиотечного набора данных в команде указывается имя набора данных вместе с именем раздела. Так, если загрузочный модуль отлаживаемой программы содержится в разделе SQRT библиотечного набора данных IVANOV.SORTAB.LOAD, абонент может ввести команду TESTFORT SORTAB(SQRT).

Если имя набора данных не согласуется с правилами именования в СРВ или набор данных принадлежит другому абоненту, имя набора данных вводится в апострофах. В этом случае нельзя опускать ни одну часть имени. Например, чтобы начать отладку программы из набора данных IVANOV.SQUARE.OBJ, абонент, работающий с идентификатором PETROV, должен ввести команду, указав в ней полное уточненное имя набора данных TESTFORT 'IVANOV. SQUARE.OBJ'.

Список имен наборов данных указывается, если модули отлаживаемой программы находятся в нескольких наборах данных. Например, если объектные модули MAIN, SOLVE и OUTPUT содержатся соответственно в наборах данных IVANOV.MAIN.OBJ, IVANOV.SOLVE.OBJ и IVANOV.OUTPUT.OBJ, можно ввести команду TESTFORT (MAIN, SOLVE, OUTPUT). Имена в списке разделяются пробелами или запятыми. При использовании в списке одного имени скобки можно опустить.

Ключевой операнд LIB содержит имена личных библиотек с загрузочными модулями отлаживаемой программы. Имена в списке разделяются запятыми или пробелами. При редактировании программы указанные библиотеки просматриваются раньше, чем библиотека программ Фортрана SYS1.FORTLIB. Имя этой библиотеки в операнде LIB задавать не требуется.

Ключевой операнд SOURCE определяет имена наборов данных, содержащих исходный текст отлаживаемой программы. Этот операнд используется только в том случае, если абонент предполагает в сеансе отладки выводить на АП предложения исходной программы. Каждый набор данных может содержать тексты одного или нескольких программных модулей и иметь последовательную или библиотечную организацию. В списке допускается не более шести таких наборов данных. Подполе операнда SOURCE или весь операнд SOURCE может отсутствовать, если в позиционном операнде указаны только имена наборов данных с объектными модулями. В этом случае происходит автоматическое распределение наборов данных с исходной программой. Их имена строятся из имен наборов данных с объектными модулями, заданными в позиционном операнде. Пусть исходные и объектные модули программы содержатся соответственно в наборах данных с именами IVANOV.

SQUARE.FORT и IVANOV.SQUARE.OBJ. Если при отладке программы абонент планирует обращение к набору данных с исходной программой, он может ввести любую из трех команд:

```
TESTFORT SQUARE SOURCE(SQUARE)
TESTFORT SQUARE SOURCE
TESTFORT SQUARE
```

Механизм автоматического распределения действует также и в случае задания позиционного операнда в виде списка. Так, по любой из двух команд

```
TESTFORT (MAIN,SOLVE,OUTPUT) SOURCE
TESTFORT (MAIN,SOLVE,OUTPUT)
```

абоненту, работающему с идентификатором IVANOV, во время отладки программы становятся доступными наборы данных с исходной программой IVANOV.MAIN.FORT, IVANOV.SOLVE.FORT и IVANOV.OUTPUT.FORT.

Если в списке отлаживаемых модулей содержится имя хотя бы одного набора данных в формате загрузочного модуля, в операнде SOURCE следует указать имена всех наборов данных с исходной программой, необходимых абоненту в сеансе отладки. Если абонент не планирует обращение к тексту исходной программы, рекомендуется указать операнд NOSOURCE.

Ключевой операнд PRINT определяет набор данных печати для вывода отладочной информации по подкомандам LIST, LISTBRKS, TRACE и WHERE. Например, по команде TESTFORT SQUARE NOSOURCE PRINT(DEBUG), введенной абонентом, работающим с идентификатором IVANOV, данные печати сохраняются в последовательном наборе данных с именем IVANOV.DEBUG.LIST. По команде TESTFORT SORTAB NOSOURCE PRINT(SORTAB(DEBUG)), введенной тем же абонентом, отладочная информация сохранится в разделе DEBUG библиотечного набора данных IVANOV.SORTAB.LIST.

Если подполе операнда или весь операнд PRINT опущен, Диалоговый отладчик распределяет для печати последовательный набор данных, которому присваивает имя: идентификатор абонента.SYS1.TESTFORT.LIST. Так, по команде TESTFORT SQUARE SOURCE PRINT, введенной абонентом, работающим с идентифи-

катором IVANOV, набор данных печати получит имя IVANOV.SYS1.TESTFORT.LIST.

Набор данных печати полезно использовать при выводе отладочной информации большого объема для последующего ее анализа за рабочим столом. Если же абонент планирует выводить отладочную информацию только на АП, рекомендуется задать операнд NOPRINT, чтобы исключить автоматическое распределение набора данных печати.

Ключевые операнды RES и NORES влияют на время ответа системы на запросы абонента и на размер области основной памяти, необходимой для диалоговой отладки программы. По операнду RES все модули Диалогового отладчика загружаются в основную память и остаются в ней в течение всей отладки. По операнду NORES в основную память загружается на постоянное хранение только часть модулей Диалогового отладчика. Остальные модули вызываются в основную память в процессе отладки. В связи с этим время ответа на запрос абонента увеличивается, но область основной памяти, используемая при отладке, уменьшается. Для отладки программы из 200 предложений в режиме NORES требуется 142 Кбайт основной памяти, в то время как в режиме RES — 166 Кбайт.

По команде TESTFORT выполняется распределение наборов данных и редактирование отлаживаемой программы. Автоматически распределяются наборы данных с объектными и загрузочными модулями, наборы данных с исходной программой, набор данных печати, набор данных SYS1.FORTLIB, содержащий библиотеку программ Фортрана, и наборы данных рабочей программы, обращение к которым выполняется по номерам 5 и 6 (последние распределяются на АП). Другие наборы данных, необходимые рабочей программе, должны быть распределены абонентом до ввода команды TESTFORT, например по команде ALLOCATE. После успешного распределения наборов данных и редактирования отлаживаемой программы в основной памяти устанавливается режим ввода подкоманд и на АП выводится сообщение TESTFORT, информирующее абонента о возможности вводить подкоманды отладки.

Пусть требуется начать отладку программы, которая содержится в наборе данных IVANOV.SQUARE.OBJ. Вся отладочная информация должна выводиться на АП.

Вывод исходной программы не предполагается. Абонент, работающий с идентификатором IVANOV, должен ввести команду

TESTFORT SQUARE NOSOURCE NOPRINT RES

Начало сеанса отладки выглядит следующим образом:

```
testfort square nosource noprint res
ILM001A TESTFORT
```

Получив разрешение на ввод подкоманд, абонент может задавать отладочные действия с АП.

7.3. ЗАДАНИЕ ОТЛАДОЧНЫХ ДЕЙСТВИЙ

Отладочные действия задаются с помощью подкоманд в определенных абонентом точках рабочей программы, называемых *точками прерывания*. Точкой прерывания может быть любой оператор Фортрана, перед выполнением которого требуется приостановить программу и произвести некоторые отладочные действия. Для указания точки прерывания используется *идентификатор оператора,* который задается меткой или номером строки оператора. Если в качестве идентификатора оператора используется метка, то ей предшествует символ «/», например, /25. В зависимости от способа установки точки прерывания делятся на *безусловные*, *условные и временные*.

При достижении точки прерывания выполнение программы приостанавливается, и абонент может задавать любые отладочные действия. Отладочные действия могут производиться как для программного модуля, в котором установлена точка прерывания, так и для других программных модулей. Для определения программного модуля, к которому относятся отладочные действия, используется *программный уточнитель*. Значение программного уточнителя совпадает с именем программного модуля. В точке прерывания программный уточнитель автоматически устанавливается равным имени программного модуля, в котором приостанавливается выполнение программы. Абонент может изменить это значение и установить его равным имени того программного модуля, для которого необходимо выполнить отладочные действия.

Переменные и массивы, используемые при задании отладочных действий, идентифицируются *уточненными*

именами. Уточненное имя состоит из двух частей: имени программного модуля и символического имени переменной или массива, разделенных точкой. Например, уточненное имя MAIN.NPAGE определяет переменную NPAGE, принадлежащую программному модулю MAIN, а уточненное имя SOLVE.NPAGE определяет переменную NPAGE, принадлежащую программному модулю SOLVE. Имя программного модуля в имени переменной или массива можно не указывать, если значение программного уточнителя совпадает с именем программного модуля, для которого задаются отладочные действия.

Отладочные действия, запрашиваемые подкомандами отладки, достаточно разнообразны. Подкоманды диалоговой отладки позволяют абоненту выполнять следующие отладочные действия: планировать точки прерывания; получать сведения о запланированных точках прерывания; управлять ходом выполнения программы; получать сведения о ходе выполнения программы; распечатывать и корректировать данные; распечатывать операторы исходной программы; получать справочную информацию о подкомандах отладки; управлять выводом отладочной информации; управлять отладочными действиями; управлять обработкой ошибок, обнаруженных в рабочей программе модулями библиотеки Фортрана; завершать сеанс отладки. Список подкоманд отладки приведен в табл. 13.

При использовании подкоманд отладки необходимо учитывать их область действия. Областью действия подкоманды может быть вся отлаживаемая программа либо отдельный программный модуль. Так, областью действия подкоманд WHEN, OFFWN, LISTBRKS, TRACE, WHERE и ERROR является вся отлаживаемая программа. Для подкоманд QUALIFY, AT, OFF, GO, RUN, LISTFREQ, LIST, SET и SOURCE областью действия является один программный модуль. Для подкоманд NEXT, HELP, PURGE, IF, HALT, FIXUP и END понятие «область действия» не имеет смысла.

При использовании подкоманд, область действия которых ограничивается одним программным модулем, имеет значение состояние этого программного модуля. В зависимости от того, получил управление программный модуль или нет, он считается *активным* или *неактивным.* Модуль, получивший управление последним, является *текущим.* Например, пусть отлаживаемая про-

Таблица 13

Назначение	Имя подкоманды	Функция подкоманды
Идентификация программного модуля	QUALIFY	Устанавливает значение программного уточнителя
Планирование точек прерывания	AT	Планирует безусловные точки прерывания
	OFF	Отменяет безусловные точки прерывания
	WHEN	Планирует условные точки прерывания
	OFFWN	Отменяет условные точки прерывания
	NEXT	Планирует временную точку прерывания
Получение сведений о запланированных точках прерывания	LISTBRKS	Выводит на АП сведения о запланированных точках прерывания
Управление ходом выполнения программы	GO	Начинает или продолжает выполнение программы, приостанавливая ее в запланированных точках прерывания
	RUN	Продолжает выполнение программы, игнорируя запланированные точки прерывания
Получение сведений о ходе выполнения программы	LISTFREQ	Выводит на АП или в набор данных печати сведения о частоте выполнения операторов
	TRACE	Запрашивает вывод на АП или в набор данных печати сведений о последующих передачах управления в программе
	WHERE	Выводит на АП или в набор данных печати сообщение о местоположении оператора и сведения о предшествующих передачах управления в программе

Продолжение табл. 13

Назначение	Имя подкоманды	Функция подкоманды
Распечатка и корректировка данных	LIST	Выводит на АП или в набор данных печати значения переменных и элементов массивов
	SET	Присваивает значения переменным и элементам массивов
Распечатка текста исходной программы	SOURCE	Выводит на АП предложения исходной программы
Получение справочной информации о подкомандах	HELP	Выводит на АП справочную информацию о подкомандах команды TESTFORT
Управление выводом отладочной информации	PURGE	Прекращает вывод отладочной информации на АП
Управление отладочными действиями	IF	Обеспечивает выполнение отладочного действия в зависимости от условия, заданного абонентом
	HALT	Передает управление абоненту для выполнения отладочных действий
Управление обработкой ошибок, обнаруженных в программе	ERROR	Устанавливает режимы обработки ошибок
	FIXUP	Выполняет корректирующее действие в случае ошибки
Завершение сеанса отладки	END	Прекращает отладку. Устанавливает режим ввода команд СРВ

грамма состоит из головного модуля MAIN и программных модулей SOLVE и OUTPUT. После того как модуль SOLVE получит управление из модуля MAIN, он становится активным и текущим, модуль MAIN является только активным, а модуль OUTPUT — неактивным. Как только модуль SOLVE вернет управление модулю MAIN, он перестает быть активным, текущим становится модуль MAIN.

Подкоманды QUALIFY, AT, OFF, LISTFREQ и SOURCE можно использовать как для активных, так и неактивных программных модулей, подкоманды LIST и SET — только для активных программных модулей, а подкоманды GO и RUN — только для текущего программного модуля.

После ввода команды TESTFORT все модули отлаживаемой программы являются неактивными. В этот момент можно вводить подкоманды, которые действуют на неактивные модули. Например, по подкоманде WHEN можно запланировать условные точки прерывания, по подкоманде ERROR — установить режимы обработки ошибок, а по подкоманде TRACE — запросить вывод сведений о передачах управления в программе. Команда TESTFORT устанавливает значение программного уточнителя равным имени головного модуля MAIN. Поэтому все подкоманды, которые действуют в пределах одного модуля, будут относиться к модулю MAIN. Так, по подкоманде AT можно запланировать безусловные точки прерывания в головном модуле, а по подкоманде SOURCE распечатать фрагменты исходного текста головного модуля MAIN. Чтобы задать эти же действия для других программных модулей, необходимо по подкоманде QUALIFY установить новый программный уточнитель.

7.4. ИДЕНТИФИКАЦИЯ ПРОГРАММНОГО МОДУЛЯ

Отладочные действия, планируемые абонентом в точке прерывания, могут относиться к различным программным модулям. Идентификация программного модуля, для которого запрашиваются последующие отладочные действия, производится с помощью подкоманды QUALIFY. Подкоманда QUALIFY имеет формат:

{ QUALIFY } [имя программного модуля]
{ Q }

Операнд определяет имя программного модуля, для которого предполагается выполнение отладочных действий. По подкоманде QUALIFY программный уточнитель устанавливается равным имени указанного программного модуля. Все подкоманды, введенные после подкоманды QUALIFY, относятся к этому программному модулю. Программный уточнитель, заданный подкомандой QUALIFY, действует до тех пор, пока не будет введена новая подкоманда QUALIFY или одна из подкоманд GO, RUN или FIXUP. По подкомандам GO, RUN и FIXUP программный уточнитель устанавливается равным имени текущего программного модуля.

Если подкоманда QUALIFY вводится без операнда, на АП выводится сообщение

ILM8011 QUALIFICATION IS имя программного модуля

содержащее значение программного уточнителя, действующего в данный момент.

Пример. Пусть выполняется отладка программы, которая состоит из головного модуля MAIN и подпрограмм SOLVE и OUTPUT. В ходе выполнения программы происходит зацикливание. Желая определить место зацикливания программы, абонент запрашивает прерывание по сигналу ВНИМАНИЕ и вводит подкоманду QUALIFY. Если в этот момент выполняется подпрограмма SOLVE, протокол отладки выглядит следующим образом:

```
* * *
% (Запрос на прерывание по сигналу ВНИМАНИЕ)
!
ILM001A TESTFORT
qualify
ILM8011 QUALIFICATION IS SOLVE
ILM001A TESTFORT
```

7.5. ПЛАНИРОВАНИЕ ТОЧЕК ПРЕРЫВАНИЯ

Диалоговая отладка за АП представляется в виде последовательности действий, выполняемых в точках прерывания, в которых абонент может проверить промежуточные результаты вычислений, при необходимости скорректировать их и продолжить выполнение программы. Точку прерывания можно установить перед любым оператором. Существует несколько способов организации точек прерывания. При первом способе абонент за-

дает в подкоманде идентификаторы операторов (номера строк или метки операторов), перед которыми требуется приостановить выполнение программы. При втором способе (когда абонент затрудняется явно указать операторы) он может определить условие, при выполнении которого необходимо организовать точку прерывания. Точки прерывания, установленные первым способом, называются безусловными, вторым способом — условными. Безусловные и условные точки прерывания позволяют организовать приостановку программы при каждом выполнении заданного оператора и соблюдении указанного условия. Третий способ позволяет приостановить выполнение программы перед оператором только один раз. Такие точки прерывания называются временными.

7.5.1. Безусловные точки прерывания

Для организации безусловных точек прерывания в программном модуле предназначена подкоманда AT. Одной подкомандой AT можно задать несколько точек прерывания. Подкоманда AT имеет формат:

```
AT { идентификатор оператора                              }
   { идентификатор оператора:идентификатор оператора      }
   { (список идентификаторов операторов)                  }
   [(список подкоманд)]
   [COUNT(n)]
   [ NOTIFY   ]
   [ NONOTIFY ]
```

Первый операнд указывает операторы, в которых должны быть организованы точки прерывания. Например, по подкоманде AT 17 точка прерывания планируется в операторе с номером строки 17, а по подкоманде AT /17 точка прерывания планируется в операторе с меткой 17.

Если абонент планирует в программном модуле сразу несколько точек прерывания, он может задать их в виде списка. Элементы списка разделяются запятыми или пробелами, а список заключается в скобки. Например, чтобы организовать точки прерывания в операторах с метками 16, 17, 18 и в операторах с номерами строк 350, 360, 370, можно ввести подкоманду AT (/16, /17,/18,350,360,370). Если операторы с метками 16, 17 и 18 следуют в программном модуле друг за другом, в

подкоманде можно указать два диапазона: AT (/16:/18, 350:370).

После обработки подкоманды AT выводится режимное сообщение и управление передается абоненту. При неоднократном задании точек прерывания протокол сеанса отладки выглядит так:

```
at /17
ILM001A TESTFORT
at (/16,/18,220,250)
ILM001A TESTFORT
at (320:380)
ILM001A TESTFORT
```

Как только при выполнении программы управление передается оператору, объявленному точкой прерывания, программа приостанавливается. Абонент получает уведомление о точке прерывания и об установке режима ввода подкоманд. Уведомление о точке прерывания выводится в виде сообщения:

```
ILM300I AT: номер строки[/метка]IN имя программного модуля
```

Если в приведенном примере управление получит оператор с меткой 17, расположенный в головном модуле MAIN в строке с номером 180, на АП выведется информация:

```
ILM300I AT: 180/17 IN MAIN
ILM001A TESTFORT
```

Режимное сообщение TESTFORT информирует абонента о возможности вводить подкоманды. Например, с помощью подкоманды LIST можно распечатать некоторые данные и в случае необходимости скорректировать их по подкоманде SET. С помощью подкоманды AT можно задавать новые точки прерывания.

Когда абонент заранее знает, какие действия он хотел бы выполнить в указанных точках прерывания, он может задать эти действия в операнде «список подкоманд». Например, если перед оператором с меткой 17 абоненту необходимо распечатать значение переменной NPAGE, он может ввести подкоманду AT /17 (LIST NPAGE). Перед тем как управление получит оператор с меткой 17, на АП выведется информация:

```
ILM300I AT: 180/17 IN MAIN
ILM840I NPAGE=       2
ILM001A TESTFORT
```

Здесь содержатся уведомление о точке прерывания, данное, запрошенное подкомандой LIST, и режимное сообщение.

Список может содержать любые подкоманды (кроме подкоманды HELP) и должен заключаться в скобки. Подкоманды в списке разделяются символом «;», например, AT /17 (LIST NPAGE;SET NPAGE=20).

Если в подкоманде задается несколько точек прерывания, список подкоманд будет выполняться в каждой из этих точек.

Использование в списке подкоманды GO или RUN позволяет после запланированной последовательности отладочных действий продолжить выполнение программы, не передавая управление абоненту. Ключевые операнды NOTIFY и NONOTIFY соответственно разрешают или запрещают вывод уведомлений о таких точках прерывания. Например, по подкоманде

```
AT /17 (LIST NPAGE;SET NPAGE=20;GO) NOTIFY
```

абонент получит уведомление о точке прерывания и выходную информацию, запрошенную подкомандой LIST. Операнд NOTIFY задавать не обязательно, так как он предполагается по умолчанию. По подкоманде с операндом NONOTIFY

```
AT /17 (LIST NPAGE;SET NPAGE=20;GO) NONOTIFY
```

абонент получит только выходную информацию, запрошенную подкомандой LIST. Уведомление о точке прерывания не выводится. В обоих случаях после вывода отладочной информации выполнение программы продолжается. Абонент должен помнить, что операнды NOTIFY и NONOTIFY имеет смысл использовать только в таких подкомандах AT, которые содержат в списке подкоманд подкоманды GO или RUN. В остальных случаях эти операнды не оказывают никакого действия.

Точки прерывания организуются в заданных операторах столько раз, сколько раз указанные операторы получают управление. Если оператор выполняется в цикле, механизм организации точек прерывания может вызвать вывод избыточной информации. Ключевой операнд COUNT позволяет управлять частотой установки точек прерывания в таких операторах. Используя этот операнд, абонент может получать точки прерывания после *n*-кратного выполнения оператора. Параметр *n* задается в виде целого десятичного положительного числа.

По умолчанию предполагается значение *n*, равное 1. Действие операнда COUNT распространяется на все точки прерывания, заданные в подкоманде AT. Например, чтобы получить управление перед выполнением оператора тела цикла с меткой 25 на каждом десятом повторении цикла, необходимо ввести подкоманду AT /25 COUNT(10).

Точки прерывания можно планировать для любых программных модулей, даже если они не являются активными. Чтобы организовать точки прерывания в программном модуле, отличном от текущего, необходимо предварительно ввести подкоманду QUALIFY, задав в ней имя требуемого модуля. При планировании точек прерывания в операторах с метками 20 и 25 подпрограммы SOLVE в точке прерывания головного модуля MAIN протокол сеанса отладки выглядит так:

```
ILM3001 AT : 110 IN MAIN
ILM001A TESTFORT
qualify solve
ILM001A TESTFORT
at (/20,/25)
ILM001A TESTFORT
```

Точки прерывания устанавливаются в заданных операторах до тех пор, пока абонент не отменит их по подкоманде OFF. Подкоманда OFF имеет формат:

OFF [идентификатор оператора
идентификатор оператора:идентификатор оператора
(список идентификаторов операторов)]

Операнд указывает операторы, в которых необходимо удалить безусловные точки прерывания. Если в подкоманде OFF операнд отсутствует, удаляются все безусловные точки прерывания в текущем программном модуле. Точки прерывания можно удалить в любом программном модуле, отличном от текущего, если предварительно ввести подкоманду QUALIFY, указав в ней имя требуемого модуля.

7.5.2. Условные точки прерывания

Для организации условных точек прерывания используется подкоманда WHEN. Подкоманда WHEN имеет формат:

{ WHEN / WN } имя условия [условие]

Первый операнд указывает имя, идентифицирующее условие. Имя может содержать от одного до четырех буквенно-цифровых символов, первым символом должна быть буква. Второй операнд определяет условие, которое может задаваться одной из следующих форм:

а) арифметическим выражением вида:

{ имя переменной
имя элемента массива }

б) отношением вида:

({ [—] имя переменной
[—] имя элемента массива
[—] константа } операция отношения
{ [—] имя переменной
[—] имя элемента массива
[—] константа })

Операция отношения обозначается следующим образом: .EQ. или =, .GT. или >, .LT. или <, .GE. или >=, LE. или <=, .NE. или ¬=. Операция отношения должна быть ограничена с двух сторон пробелами (например, A .GT. B или A = B);

в) логическим множителем вида:

([.NOT.] { имя логической переменной
имя элемента логического массива })
[¬]

После операции отрицания должен следовать, по крайней мере, один пробел (например, .NOT. LOG или ¬ LOG).

Во всех формах записи условия можно использовать константы, переменные и элементы массива любых типов, допустимых в языке Фортран. В элементе массива индексное выражение может задаваться в виде I, C или I ± C, где I — имя целой переменной, а C — целая константа. Имена переменных и элементов массивов могут быть уточненными, например SUB1.A .EQ. SUB2.B. Уточненные имена указываются в тех случаях, когда в разных программных модулях используются одинаковые имена переменных и массивов.

Действие подкоманды WHEN распространяется на модули отлаживаемой программы, протранслированные в режиме TEST. Условие, заданное в подкоманде (назовем его условием WHEN), проверяется перед выполнением каждого оператора отлаживаемых модулей. В случае выполнения условия устанавливается точка прерывания и управление передается абоненту. Абонент

получает уведомление об условной точке прерывания в виде сообщения:

```
ILM871I WNEN: имя условия IN номер строки [/метка] IN
        имя программного модуля
```

За этим сообщением следует сообщение TESTFORT.

Форма записи условия а) позволяет организовать точки прерывания при любых изменениях значения переменной или элемента массива. Например, если абоненту необходимо получать управление в случае изменения значения переменной ALPHA, он может ввести подкоманду WHEN RDA ALPHA. Имя RDA идентифицирует условие проверки значения переменной ALPHA. Контроль за изменением значений этой переменной ведется перед выполнением каждого оператора. Так, если после ввода подкоманды WHEN в головном модуле выполняются операторы

```
16 ALPHA=SQRT(BETA)                    00000170
17 WRITE(6,27) ALPHA                   00000180
```

абонент получит следующую информацию на АП:

```
ILM871I WHEN: RDA IN 180/17 IN MAIN
ILM001A TESTFORT
```

Получив управление, абонент может вводить любые подкоманды отладки. По подкоманде GO можно возобновить выполнение программы. Условие RDA будет проверяться во всех последующих операторах программы.

Форма записи условия б) позволяет организовать точки прерывания в тех случаях, когда значение отношения между переменными и элементами массива истинно. Предположим, что абоненту необходимо получать управление, когда значение переменной BETA становится равным нулю или значению переменной ALPHA. Если заданные условия именовать соответственно RDB и RDA, планирование точек прерывания выглядит следующим образом:

```
when rdb (beta .eq. 0.0)
ILM001A TESTFORT
when rda (beta .eq. alpha)
ILM001A TESTFORT
```

Как только во время выполнения программы наступает одно из этих условий (отношение принимает значение «истина»), устанавливается точка прерывания и управление передается абоненту.

Форма записи условия в) позволяет организовать

точки прерывания, если значение логического множителя истинно. Так, по подкоманде WHEN (.NOT. LOG) абонент будет получать управление в тех случаях, когда логическая переменная LOG принимает значение «ложь».

Условия, заданные в подкомандах WHEN, проверяются до тех пор, пока абонент не отменит установку условных точек прерывания.

Для отмены условных точек прерывания используется подкоманда OFFWN, которая имеет формат:

OFFWN[имя условия / (список имен условий)]

В операнде задаются имена одного или нескольких условий, проверку которых требуется прекратить. Имена условий в списке разделяются запятыми или пробелами, а список заключается в скобки. По подкоманде OFFWN без операнда прекращается проверка всех условий WHEN, определенных в отлаживаемой программе. Например, чтобы прекратить проверку условий WHEN, идентифицированных именами RDA и RDB, абонент должен ввести подкоманду OFFWN (RDA, RDB). По подкоманде OFFWN прекращается только проверка условий, а определение условий сохраняется. Чтобы возобновить, например, проверку условия RDA, в подкоманде WHEN достаточно указать только его имя: WHEN RDA. По подкоманде RUN не только прекращается проверка условий, но и удаляются определения условий.

7.5.3. Вре́менная точка прерывания

Планирование вре́менной точки прерывания производится по подкоманде NEXT. Вре́менная точка прерывания определяет оператор, который должен выполняться первым после текущего оператора. Текущим является оператор, перед которым производятся отладочные действия. Установка временной точки прерывания произойдет, как только выполнится текущий оператор и управление получит следующий оператор. Программа приостанавливается, и абоненту посылается уведомление в виде сообщения:

ILM3011 NEXT: номер строки[/метка] IN имя программного модуля

За ним следует режимное сообщение TESTFORT, позволяющее вводить подкоманды отладки.

Планирование временной точки прерывания рассмотрим на примере. Предположим, абонент получил управление в безусловной точке прерывания подпрограммы SUB1 перед оператором с меткой 5 в строке с номером 180, который имеет вид:

```
5 GO TO (10,25,30,33,39,41), K                    00000180
```

После выполнения оператора с меткой 5 управление может получить любой из операторов, указанный в операторе GO TO. Пусть к моменту выполнения оператора с меткой 5 значение переменной K равно 4. Это означает, что следующим должен выполняться оператор с меткой 33, расположенный, например, в строке с номером 250. Если абонент введет подкоманду NEXT, а затем подкоманду GO, то планирование и установка временной точки прерывания на протоколе отладки выглядят следующим образом:

```
ILM300I AT: 180/5 IN SUB1
ILM001A TESTFORT
next
ILM001A TESTFORT
go
ILM301I NEXT: 250/33 IN SUB1
ILM001A TESTFORT
```

По подкоманде NEXT временная точка прерывания только планируется. Установка этой точки производится после того, как программа получит управление по подкоманде GO. Однако подкоманда GO необязательно должна следовать сразу же за подкомандой NEXT. Между ними могут быть введены другие подкоманды, не возобновляющие выполнение программы (например, LIST, SET).

В тех случаях, когда абонент заранее планирует временную точку прерывания, он может указать подкоманду NEXT в списке подкоманды AT. Так, в рассмотренном примере абонент может ввести подкоманду AT вида:

```
AT /5 (NEXT;LIST K;GO)
```

7.5.4. Получение сведений о запланированных точках прерывания

По подкоманде LISTBRKS (она состоит только из имени) абонент может получить сведения о точках прерывания, запланированных в отлаживаемой программе.

Эти сведения выводятся в виде двух списков: списка безусловных точек прерывания и списка условий WHEN.

Список безусловных точек прерывания содержит сведения о точках прерывания, запланированных во всех программных модулях отлаживаемой программы с помощью подкоманд AT, и идентифицируется заголовком: ILM950I CURRENT BREAKPOINTS (Текущие точки прерывания). Для каждой точки прерывания в списке присутствует строка вида

```
{ ILM952I }номер строки [/метка]
{ ILM953I }
        IN имя программного модуля [COUNT(n)]
```

Сообщение содержит номер строки оператора, метку оператора (если она задана), имя программного модуля и значение счетчика, заданного в операнде COUNT.

Если в отлаживаемой программе точки прерывания не планировались или все они удалены с помощью подкоманды OFF, после заголовка выводится строка ILM943I NONE.

Список условий WHEN содержит сведения обо всех условиях, определенных в отлаживаемой программе. Список идентифицируется заголовком: ILM955I CURRENT WHEN CONDITIONS (Действующие условия WHEN). Для каждого условия WHEN в списке содержится строка следующего вида:

```
ILM957I имя условия  { ON  }
                     { OFF }
```

Значение ON или OFF указывает, что проверка условия соответственно разрешена или запрещена к данному моменту. Если условия WHEN в отлаживаемой программе не определены, после заголовка выводится строка ILM943I NONE.

Пример. В отлаживаемой программе, состоящей из головного модуля MAIN и подпрограммы DISK, определены два условия WHEN, идентифицируемые именами RDS и WTO. В головном модуле MAIN по подкоманде AT точка прерывания установлена в операторе с меткой 80 (номер строки 250), а в подпрограмме DISK — в операторах с номерами строк 110—220. Выполнение программы приостановлено в подпрограмме DISK в строке с номером 180. Если к этому моменту проверка условия WTO прекращена, по подкоманде LISTBRKS на АП выведется информация:

```
listbrks
ILM950I CURRENT BREAKPOINTS
ILM952I 250/80 IN MAIN
ILM952I 110 IN DISK
   . . .
ILM952I 220 IN DISK
ILM955I CURRENT WHEN CONDITIONS
ILM957I RDS ON
ILM957I WTO OFF
ILM001A TESTFORT
```

7.6. УПРАВЛЕНИЕ ХОДОМ ВЫПОЛНЕНИЯ ПРОГРАММЫ

Для управления последовательностью выполнения операторов в программе предназначены подкоманды GO и RUN. Эти подкоманды имеют формат:

```
GO [идентификатор оператора]
RUN[идентификатор оператора]
```

Обе подкоманды позволяют возобновить выполнение программы с точки прерывания или с любого другого оператора текущего программного модуля. При возобновлении выполнения программы с точки прерывания идентификатор не указывается. Например, по подкоманде GO выполнение программы будет продолжено с точки прерывания, а по подкоманде GO /25 — с оператора с меткой 25. После ввода команды TESTFORT выполнение программы можно начать только с первого оператора головного модуля. Для этого необходимо ввести подкоманду GO или RUN без операнда. Подкоманды GO и RUN оказывают различные действия на отлаживаемую программу. По подкоманде GO производится приостановка программы во всех запланированных точках прерывания. После возобновления выполнения программы по подкоманде RUN точки прерывания игнорируются. Приостановка программы происходит только при возникновении ошибки или в результате прерывания выполнения программы по сигналу ВНИМАНИЕ. В тех случаях, когда подкомандами GO и RUN абонент продолжает выполнение программы не с точки прерывания, он должен правильно выбрать оператор, которому передает управление внутри текущего модуля. Например, входы внутрь тела цикла могут привести к непредсказуемым результатам. Запрещается также передача управления из одного программного модуля в другой. Подкоманда GO или RUN, введенная после подкоманды QUALIFY, отменяет заданный в ней про-

граммный уточнитель и устанавливает его равным имени текущего программного модуля. Не разрешается использовать подкоманды GO и RUN после завершения программы по оператору STOP. Сообщение о завершении выполнения программы имеет вид:

```
ILM306I PROGRAM UNDER TESTFORT TERMINATED
        NORMALLY
```

(Программа завершилась нормально)

В ответ на это сообщение можно вводить любые подкоманды, кроме подкоманд GO и RUN. Когда абонент планирует повторное выполнение программы, рекомендуется установить точки прерывания перед операторами STOP. Подкоманда GO или RUN, введенная в таких точках прерывания, позволяет возобновить выполнение программы с любого оператора.

Пример. Приводится фрагмент сеанса отладки программы, иллюстрирующий возможности подкоманд GO и RUN.

Шаг 1. Установка режима диалоговой отладки программы, состоящей из головного модуля MAIN и подпрограмм SOLVE и OUTPUT, содержащихся в наборе данных IVANOV.SQUARE.OBJ.

```
testfort square nosource noprint
ILM001A TESTFORT
```

Шаг 2. Планирование точек прерывания в модулях MAIN и SOLVE.

```
at (/17, 180)
ILM001A TESTFORT
qualify solve
ILM001A TESTFORT
at 230
ILM001A TESTFORT
```

Шаг 3. Запуск программы.

```
go
```

Выполнение программы приостанавливается в точке прерывания:

```
ILM300I AT: 230 IN SOLVE
ILM001A TESTFORT
```

Шаг 4. Распечатка значения переменной DISK.

```
list disk
ILM840I DISK= 1
ILM001A TESTFORT
```

Шаг 5. Так как результат вычисления, полученный в шаге 4, оказался верным, абонент продолжает выполнение программы без приостановки в запланированных точках прерывания.

```
run
ILM306I PROGRAM UNDER TESTFORT TERMINATED
        NORMALLY
ILM001A TESTFORT
```

7.7. ПОЛУЧЕНИЕ СВЕДЕНИЙ О ХОДЕ ВЫПОЛНЕНИЯ ПРОГРАММЫ

В тех случаях, когда в программе происходит зацикливание или возникает ошибка, абоненту может потребоваться информация о ходе выполнения отдельных логических частей программы. Для получения таких сведений он может использовать подкоманды TRACE, WHERE и LISTFREQ.

По подкоманде TRACE выдается последовательность передач управления между операторами внутри программного модуля, а также информация о входах в программные модули и выходах из них. Прослеживание передач управления начинается с оператора, указанного в последующей подкоманде GO или RUN, и продолжается до тех пор, пока не будет введена подкоманда TRACE с операндом OFF. Подкоманда TRACE имеет формат:

$$\left\{\begin{matrix}\text{TRACE}\\ \text{T}\end{matrix}\right\}\left[\begin{matrix}\underline{\text{STMT}}\\ \underline{\text{ENTRY}}\\ \text{OFF}\end{matrix}\right]\ [\text{PRINT}]$$

По операнду ENTRY выводятся сведения о входах в программные модули и выходах из них. Указав операнд STM, дополнительно можно получить сведения о последующих передачах управления в программе.

По операнду OFF прекращается вывод сведений, запрошенных предыдущей подкомандой TRACE.

Операнд PRINT позволяет направить отладочную информацию в набор данных печати. Имя набора данных указывается в команде TESTFORT. Если операнд PRINT опущен, отладочная информация выводится на АП.

Информация, выводимая по подкоманде TRACE, содержит строки четырех видов. При входе в каждый модуль-процедуру выводится строка:

ILM302I TRACE: имя модуля-процедуры ENTERED

Например, после входа в подпрограмму SUB1 на АП выведется строка

ILM302I TRACE: SUB1 ENTERED

Если модуль-процедура имеет несколько точек входа, строка вывода имеет вид:

ILM303I TRACE: имя-модуля-процедуры ENTERED AT
имя точки входа

Например, после входа в точку ENTRY1 подпрограммы SUB1 на АП выведется строка

ILM303I TRACE SUB1 ENTERED AT ENTRY1

После выхода из каждого модуля-процедуры выводится строка

ILM305I TRACE: RETURN FROM имя модуля-процедуры

Например, после выхода из подпрограммы SUB1 на АП выведется строка

ILM305I TRACE: RETURN FROM SUB1

В случае передачи управления внутри модуля выводится строка

ILM304I TRACE: FROM номер строки[/метка] TO
номер строки [/метка]

Если абонент ввел подкоманду TRACE с операндом STMT, то отладочная информация может содержать строки всех четырех видов. Если абонент ввел подкоманду TRACE с операндом ENTRY, отладочная информация может содержать строки первых трех видов.

По подкоманде TRACE абонент запрашивает вывод информации о последующем ходе выполнения программы, начиная с момента запроса. Чтобы получить сведения о ходе выполнения программы, предшествующем моменту запроса, а не после него, следует использовать WHERE. Подкоманда WHERE обеспечивает вывод информации о местоположении приостановки выполнения программы.

{ WHERE } [TRBACK][FLOW] [PRINT]
{ W } [NOTRBACK]

Если абонент укажет операнд TRBACK, он получит сообщение о местоположении точки прерывания и список вызываемых программных модулей. По операнду NOTRBACK выведется только сообщение о местоположении точки прерывания.

Если абонент укажет операнд FLOW, он получит сообщение о местоположении точки прерывания и сведения о последних десяти передачах управления в программе.

По операнду PRINT отладочная информация выводится в набор данных печати. Имя набора данных указывается в команде TESTFORT. Если операнд PRINT опущен, вывод производится на АП.

Сообщение о местоположении точки прерывания имеет вид:

ILM995I WHERE: номер строки[/метка] IN имя программного модуля

Сообщение содержит номер строки оператора, метку оператора (если она указана в операторе) и имя программного модуля.

Сведения о вызываемых программных модулях выводятся в виде списка, в котором для каждого вызываемого программного модуля содержится строка вида:

ILM991I имя вызываемого модуля CALLED BY имя вызывающего модуля AT номер строки[/метка]

Строки списка выводятся в порядке, обратном порядку обращения к программным модулям. В списке присутствует информация только о программных модулях, протранслированных в режиме TEST.

Если приостановка программы произошла в головном модуле, выводится сообщение об отсутствии вызываемых программных модулей:

ILM992I NO SUBROUTINES CALLED

Сведения о передачах управления выводятся также в виде списка. Для каждой передачи управления в списке содержатся две строки:

ILM997I TO: номер строки[/метка] IN имя программного модуля
ILM996I FROM: номер строки[/метка] IN имя программного модуля

Первая строка содержит идентификатор оператора, которому передается управление от оператора, идентифицированного нижней строкой. Строки списка выводятся в порядке, обратном порядку выполнения операторов. Список содержит сведения не более чем о десяти последних передачах управления.

Пример 1. Выполняется отладка программы, состоящей из головного модуля MAIN и подпрограмм SOLVE и OUTPUT. Пусть точка прерывания установлена в подпрограмме OUTPUT. Если абонент введет подкоманду WHERE с операндом TRBACK, протокол отладки может выглядеть следующим образом:

```
where trback
ILM995I WHERE: 460 IN OUTPUT
ILM991I OUTPUT CALLED BY MAIN AT 130
ILM001A TESTFORT
```

Пример 2. Выполняется отладка программы, состоящей из программных модулей MAIN и RESTAB. Допустим, что абоненту необходимо прервать выполнение программы, чтобы выполнить некоторые отладочные действия. Последовательность отладочных действий зависит от того, в каком программном модуле, MAIN или RESTAB, произойдет прерывание. Для определения модуля, выполнение которого приостановлено по сигналу ВНИМАНИЕ, абонент вводит подкоманду WHERE. Протокол отладки выглядит так:

```
***
% (Запрос на прерывание по сигналу ВНИМАНИЕ)
!
ILM001A TESTFORT
where
ILM995I WHERE: 340 IN RESTAB
ILM001A TESTFORT
```

Из протокола следует, что прерывание произошло в подпрограмме RESTAB перед выполнением оператора в строке с номером 340.

Пример 3. Предположим, что абонента интересуют значения массива ARRAY, состоящего из 200 элементов, до выполнения операторов с метками 5 и 17. Если абонент введет подкоманду

```
AT (/5,/17) (LIST ARRAY PRINT;GO) NOTIFY
```

то при многократном ее выполнении ему трудно будет разобраться в отладочной информации, накопленной в наборе данных печати. Если же абонент введет подкоманду

```
AT (/5,/17) (WHERE PRINT;LIST ARRAY PRINT;GO) NOTIFY
```

то в наборе данных печати каждый вывод массива ARRAY будет идентифицироваться заголовком в виде сообщения о местоположении точки прерывания (уведомление об установлении точки прерывания по подкоманде AT выводится только на АП).

Подкоманда LISTFREQ предоставляет абоненту возможность получать сведения о частоте выполнения указанных им операторов программы. Такие сведения могут быть полезны при определении полноты проверки

алгоритма программы в конкретном сеансе отладки. Подкоманда LISTFREQ имеет формат:

```
LISTFREQ [ идентификатор оператора
         [ идентификатор оператора:
                идентификатор оператора
         [ (список идентификаторов операторов) ]
[ZEROFREQ] [PRINT]
```

Первый операнд определяет операторы, для которых запрашиваются сведения о частоте их выполнения. Указанные операторы должны принадлежать одному программному модулю. Чтобы получить сведения о частоте использования всех операторов данного модуля, достаточно ввести только имя подкоманды. Сведения выводятся в виде списка, который начинается заголовком:

```
ILM945I STATEMENTS IN имя программного модуля
        FREQUENCY
        (Частота выполнения операторов в программном модуле)
```

Для каждого оператора в списке присутствует строка вида:

```
ILM9471 номер строки[/метка] число
```

Сообщение содержит номер строки, метку оператора (если она есть) и число, которое указывает, сколько раз выполнялся данный оператор. Для получения сведений об операторах, которые не выполнялись, используется операнд ZEROFREQ. По подкоманде LISTFREQ ZEROFREQ выводится список, который идентифицируется заголовком

```
ILM940I ZERO — FREQUENCY STATEMENTS IN
    имя программного модуля
        (Невыполняемые операторы в программном модуле)
```

Список содержит только номера строк и метки (если они есть) тех операторов, которые к данному моменту еще не выполнялись. Операнд ZEROFREQ позволяет определить, есть ли в модуле непроверенные или неработающие фрагменты. Если такие фрагменты отсутствуют, абонент получит уведомление в виде сообщения:

```
ILM943I NONE
```

Операнд PRINT позволяет направить отладочную информацию в набор данных печати, имя которого указывается в команде TESTFORT. Если операнд опущен, информация выводится на АП.

Подкоманду LISTFREQ можно использовать для любого программного модуля. Чтобы получить сведения о программном модуле, отличном от текущего, необходимо предварительно ввести подкоманду QUALIFY.

Пример 4. Требуется получить сведения о частоте выполнения операторов программы, состоящей из головного модуля MAIN и подпрограммы SOLVE. Выполнение программы приостановлено в точке прерывания головного модуля MAIN.

Шаг 1. Запрос на вывод сведений о частоте выполнения отдельных операторов модуля MAIN.

```
listfreq  (/17,180)
ILM945I STATEMENTS IN MAIN FREQUENCY
ILM947I 170/17                1
ILM947I 180                   1
ILM001A TESTFORT
```

Шаг 2. Запрос на вывод сведений о частоте выполнения всех операторов подпрограммы SOLVE.

```
qualify solve
ILM001A TESTFORT
listfreq
ILM945I STATEMENTS IN SOLVE FREQUENCY
ILM947I 220                   1
ILM947I 230                   1
ILM947I 240/50                1
. . .
ILM947I 390                   0
ILM001A TESTFORT
```

Шаг 3. Запрос на вывод сведений о частоте выполнения всех операторов подпрограммы SOLVE в набор данных печати.

```
listfreq print
ILM001A TESTFORT
```

После ввода команды LISTFREQ перед выводом режимного сообщения организуется пауза, во время которой производится вывод в набор данных печати.

Шаг 4. Запрос на вывод сведений об операторах, которые не выполнялись в головном модуле.

```
qualify main
ILM001A TESTFORT
listfreq zerofreq
ILM940I ZERO-FREQUENCY STATEMENTS IN MAIN
ILM943I NONE
ILM001A TESTFORT
```

7.8. РАСПЕЧАТКА И КОРРЕКТИРОВКА ДАННЫХ

Распечатка и корректировка данных являются наиболее употребительными средствами отладки, так как позволяют в процессе отладки следить за промежуточ-

ными значениями данных и изменять их в случае необходимости. Распечатка данных производится с помощью подкоманды LIST. По одной подкоманде LIST можно распечатать отдельные либо все данные программного модуля. Вывод может быть направлен на АП или в набор данных печати. Подкоманда LIST имеет формат:

$$\left\{\begin{matrix}\text{LIST}\\ \text{L}\end{matrix}\right\} \left\{\begin{matrix}\text{имя}\\ \text{имя:имя}\\ \text{(список имен)}\\ *\end{matrix}\right\} [\text{DUMP}[(\text{код})]]\ [\text{PRINT}]$$

Первый операнд — позиционный. Он определяет имена переменных, элементов массивов и массивов, значения которых требуется распечатать. Элементы в списке разделяются запятыми или пробелами, а список заключается в скобки. Чтобы распечатать все данные программного модуля, в подкоманде LIST достаточно указать символ «*». При выборе границ диапазона необходимо учитывать расположение данных в основной памяти. Неправильное задание границ диапазона может привести к выводу избыточной информации. Сведения о расположении данных в основной памяти содержатся в распечатке распределения памяти, выдаваемой транслятором Фортран SE.

Операнд DUMP позволяет управлять форматом вывода данных. Задаваемое значение параметра «код» должно согласовываться с типом и длиной данного. Ниже приведены форматы вывода в зависимости от параметра «код».

Код	Тип, длина операнда
Z	Шестнадцатеричный
L1	Логический, байт
L4 или L	Логический, слово
I2	Целый, полуслово
I4 или I	Целый, слово
R4 или R	Вещественный, слово
R8	Вещественный, двойное слово
C8	Комплексный, два слова
C16	Комплексный, два двойных слова
H	Текстовый

Если в операнде DUMP параметр «код» опущен, по умолчанию предполагается формат Z. Операнд DUMP игнорируется, если в позиционном операнде задан диапазон имен или *.

Операнд DUMP может не указываться. В этом случае формат вывода зависит от способа задания позицион-

ного операнда. Если в позиционном операнде указано имя, список имен или *, данные выводятся в соответствии с их типом по коду преобразования G. Если задан диапазон имен, то данные выводятся в шестнадцатеричном виде.

По операнду PRINT данные выводятся в набор данных печати. Имя набора данных указывается в команде TESTFORT. Если операнд PRINT опущен, информация выводится на АП.

Пример 1. Иллюстрируются возможности подкоманды LIST для вывода значений вещественных переменных X1REAL и X1IMAG.

Запрос на вывод значений переменных в соответствии с их типом:

```
list (x1real x1imag)
ILM840I X1REAL = —2.00000000
ILM840I X1IMAG = 1.41421318
ILM001A TESTFORT
```

Запрос на вывод значений переменных в шестнадцатеричном формате:

```
list (x1real x1imag) dump
ILM850I 05AB00  C1200000
ILM850I 05AB04  4116A09E
ILM001A TESTFORT
```

Запрос на вывод значений переменных с указанием формата в операнде DUMP:

```
list (x1real x1imag)  dump(r)
ILM850I 05AB00     —0.200000000E+01
ILM850I 05AB04      0.141421318E+01
ILM001A TESTFORT
```

Пример 2. Иллюстрируются возможности подкоманды LIST для вывода значений массива ARRAY целого типа, состоящего из 10 элементов, сначала на АП, а затем в набор данных печати.

```
list array
ILM840I ARRAY =          1,      2,      3.      4,      5,
                         6,      7,      8,      9,     10
ILM001A TESTFORT
list array print
ILM001A TESTFORT
```

После ввода подкоманды LIST ARRAY PRINT перед появлением режимного сообщения TESTFORT организуется пауза, во время которой производится вывод в набор данных печати.

Для корректировки данных используется подкоманда SET. С помощью подкоманды SET можно изменить значение одной переменной, одного или нескольких элементов массива или всего массива. Подкоманда SET имеет формат:

{ SET } оператор присваивания
{ S }

Подкоманда содержит один позиционный операнд, который записывается в виде оператора присваивания a=b. Форма записи этого оператора несколько отличается от записи соответствующего оператора языка Фортран. Параметр a указывает имя переменной, элемента массива или массива. Параметр b указывает одно или несколько значений, которые заменяют текущие значения данных. В арифметическом операторе присваивания параметром b может быть имя переменной, элемента массива или массива, либо константы любого типа, допустимого в языке Фортран. Если a и b различных типов, то выполняется преобразование операнда b к типу операнда a по правилам языка Фортран. В логическом операторе присваивания параметром b может быть имя логической переменной, элемента массива или массива, либо логические константы .TRUE. и .FALSE. В элементе массива, используемом в параметрах a и b, индексное выражение записывается в виде I, C или I±C, где I — имя целой переменной, а C — целая константа.

Одна подкоманда SET позволяет присвоить значения в виде констант отдельным или всем элементам массива. Эти значения заключаются в скобки и разделяются между собой запятыми или пробелами.

Для присваивания одинаковых значений нескольким последовательным элементам массива можно использовать указатель повторения j*. Если после указателя повторения константа отсутствует, то соответствующие элементы массива не изменяются. Например, 3*20.5 означает, что трем элементам массива присваивается значение 20.5; 5* означает, что пять элементов массива остаются без изменения.

Пример 3.

```
SET AB=BC
```

Переменной AB присваивается значение переменной BC.

```
SET N = (1,1,1,3*,4*2)
```

Первым трем элементам массива N, состоящего из 10 элементов, присваивается значение 1, следующие три элемента остаются без изменения, а последним четырем элементам массива присваивается значение 2.

SET LOG4=.TRUE.

Логической переменной LOG4 присваивается значение «истина». Указанная корректировка данных на протоколе отладки выглядит так:

```
set ab=bc
ILM001A TESTFORT
set n =(1,1,1,3*,4*2)
ILM001A TESTFORT
set log4=.true.
ILM001A TESTFORT
```

В примерах демонстрировалось использование подкоманд LIST и SET для распечатки и корректировки данных текущего программного модуля. В процессе отладки у абонента может появиться необходимость обращаться к данным других программных модулей. Диалоговый отладчик предоставляет такую возможность для активных программных модулей. После ввода команды TESTFORT (до передачи управления отлаживаемой программе) все модули считаются неактивными. В этот момент нельзя ни распечатывать, ни корректировать данные. Получив управление в запланированной точке прерывания, абонент может распечатывать и корректировать любые данные тех модулей, которые стали активными. Чтобы обратиться к данному из программного модуля, отличного от текущего, в подкомандах LIST и SET необходимо использовать уточненное имя. Так, чтобы распечатать зачение переменной NPAGE модуля MAIN в точке прерывания подпрограммы SOLVE, достаточно ввести подкоманду LIST MAIN.NPAGE. Присвоить этой переменной новое значение можно по подкоманде SET MAIN.NPAGE=20. Чтобы обратиться к нескольким переменным какого-либо модуля, имя этого модуля следует указать перед каждым именем данного. Так, чтобы распечатать значения переменных A1, B1 и C1 модуля MAIN, абонент должен ввести подкоманду LIST (MAIN.A1,MAIN.B1,MAIN.C1). При задании длинного списка такая запись вызывает большие неудобства. Чтобы сократить запись, рекомендуется использовать подкоманду QUALIFY. Например, распечат-

ку переменных A1, B1, C1, A2, B2, C2 модуля MAIN можно выполнить вводом двух подкоманд:

```
QUALIFY MAIN
LIST (A1,B1,C1,A2,B2,C2)
```

По одной подкоманде LIST можно вывести данные разных программных модулей. Так, если точка прерывания установлена в подпрограмме OUTPUT, то по подкоманде LIST (MAIN.NPAGE, SOLVE. DISC) распечатаются значения переменной NPAGE модуля MAIN и переменной DISC модуля SOLVE.

В подкоманде SET можно также использовать переменные разных программных модулей. По подкоманде SET MAIN.NPAGE = SUB1.LIST переменной NPAGE модуля MAIN присвоится значение переменной LIST модуля SUB1.

Значения, присвоенные подкомандой SET, сохраняются только на время одного сеанса отладки. Поэтому после отладки программы все необходимые изменения следует внести в наборы данных с исходной программой, используя для этой цели, например, команду EDIT.

7.9. РАСПЕЧАТКА ТЕКСТА ИСХОДНОЙ ПРОГРАММЫ

Наборы данных с текстом исходной программы, указанные в команде TESTFORT, доступны абоненту в течение всего сеанса отладки. Во время сеанса отладки абоненту нельзя использовать более шести таких наборов данных. Каждый из наборов данных может содержать тексты нескольких программных модулей. Если абоненту необходимо обратиться к тексту этих модулей, он должен использовать подкоманду SOURCE. Подкоманда SOURCE позволяет вывести на АП отдельные предложения или весь текст программного модуля из любого набора данных, доступного абоненту. Подкоманда SOURCE имеет формат:

```
{ SOURCE }  [ идентификатор предложения                          ]
{ SO     }  [ идентификатор предложения:                         ]
            [           идентификатор предложения                ]
            [ (список идентификаторов предложений)               ]
            [ *                                                  ]
```

Операнд определяет предложения программного модуля, которые требуется вывести на АП. Символ * используется для вывода оператора, в котором установлена точка прерывания. Чтобы вывести весь текст про-

граммного модуля, достаточно ввести подкоманду SOURCE без операнда.

Подкоманду SOURCE можно задавать для любого программного модуля отлаживаемой программы. При выводе текста программного модуля, отличного от текущего, необходимо предварительно ввести подкоманду QUALIFY, задав в ней имя требуемого модуля. Информация, выводимая по подкоманде SOURCE, начинается заголовком

```
ILM930I SOURCE FOR имя программного модуля
                (Исходный текст программного модуля)
```

Пример. Иллюстрируются возможности подкоманды SOURCE для вывода исходного текста головного модуля MAIN и подпрограммы SOLVE. Точка прерывания установлена в головном модуле MAIN (оператор с меткой 17 в строке с номером 110). Абонент выводит отдельные фрагменты исходного текста: оператор, в котором установлена точка прерывания, и операторы из диапазона строк 240—260 подпрограммы SOLVE.

```
source *
ILM930I SOURCE FOR MAIN
ILM935I 00110       17 STOP
ILM001A TESTFORT
qualify solve
ILM001A TESTFORT
source 240: 260
ILM930I SOURCE FOR SOLVE
ILM935I 00240 50 X1REAL=—B(2.0*A)
ILM935I 00250    X2REAL=X1REAL
ILM935I 00260    X1IMAG=SQRT(—DISC)/(2.0*A)
ILM001A TESTFORT
```

7.10. ПОЛУЧЕНИЕ СПРАВОЧНОЙ ИНФОРМАЦИИ О ПОДКОМАНДАХ ОТЛАДКИ

В процессе отладки абонент может испытывать затруднения при использовании подкоманд. В таких случаях с помощью подкоманды HELP он может запросить вывод на АП сведений о функциях, синтаксисе и операндах любой подкоманды команды TESTFORT. Подкоманда HELP имеет такой же формат, как и команда HELP.

Например, чтобы получить сведения только о функциях операндов COUNT и NOTIFY подкоманды AT, следует ввести подкоманду HELP AT FUNCTION OPERANDS(COUNT,NOTIFY). Если абонент введет под-

команду HELP AT OPERANDS(COUNT, NOTIFY), он получит описание функций и синтаксиса этих операндов. Чтобы получить всю информацию о подкоманде AT, можно ввести подкоманду HELP AT. При вводе подкоманды HELP без операндов на АП выводится список всех подкоманд команды TESTFORT и краткое описание каждой из них. Напомним, что справочную информацию о команде TESTFORT можно получить по команде HELP TESTFORT, которая должна быть введена в режиме ввода команд СРВ.

7.11. УПРАВЛЕНИЕ ВЫВОДОМ ОТЛАДОЧНОЙ ИНФОРМАЦИИ

Вывод отладочной информации производится по подкомандам LIST, LISTBRKS, LISTFREQ, SOURCE и HELP. Остановить вывод можно, используя прерывание по сигналу ВНИМАНИЕ. Действия абонента в этом случае зависят от того, каким способом был запрошен вывод: непосредственно с АП или в списке подкоманды AT.

Если подкоманда вывода поступила с АП, то достаточно вызвать прерывание по сигналу ВНИМАНИЕ. В результате прерывания вывод прекращается и абонент может вводить любые подкоманды отладки.

Если подкоманда вывода поступила из списка подкоманды AT, то, чтобы прекратить действие прерванной подкоманды, необходимо после прерывания по сигналу ВНИМАНИЕ ввести подкоманду PURGE. Дальнейшие действия зависят от абонента. Абонент может ввести пустую строку или любую подкоманду отладки. В первом случае произойдет выполнение отладочных действий, запланированных последующими подкомандами списка, во втором случае — оставшиеся невыполненными подкоманды списка игнорируются.

Пример 1. Абонент ввел подкоманду LIST ARRAY для распечатки массива ARRAY, состоящего из 200 элементов. При достижении конца экрана после вывода на АП значений части элементов массива абонент решает прекратить вывод остальных элементов массива. Действия абонента для прекращения распечатки массива ARRAY выглядят следующим образом:

```
list array
ILM840I ARRAY=   1,  2,  3,  4,  5,
                 6,  7,  8,  9, 10,
                11, 12, 13, 14, 15,
```

```
*** (Уведомление о конце экрана)
% (Запрос на прерывание по сигналу ВНИМАНИЕ.
   Вывод прекращается)
!
ILM001A TESTFORT
```

Пример 2. Пусть запрашивается вывод на АП массивов N, M и L, содержащих по 100 элементов, по подкоманде AT /17 (LIST N;LIST M;LIST L). Фрагмент сеанса отладки иллюстрирует, как можно с помощью сигнала ВНИМАНИЕ и подкоманды PURGE прекращать вывод массивов.

```
ILM300I AT: 250/17 IN MAIN
ILM840I N=      1,  2,  3,  4,  5,
                6,  7,  8,  9, 10,
                  .  .  .
*** (Уведомление о конце экрана. Абонент решает
   прекратить вывод остальных элементов массива N)
% (Запрос на прерывание по сигналу ВНИМАНИЕ)
!
ILM001A TESTFORT
purge (Действие подкоманды LIST N прекращается)
ILM001A TESTFORT
   (Ввод пустой строки, чтобы начать
   вывод массива M)
ILM840I M= 11, 12, 13, 14, 15,
           16, 17, 18, 19, 20
             .  .  .
*** (Уведомление о конце экрана. Абонент решает
   прекратить вывод остальных элементов массива M)
% (Запрос на прерывание по сигналу ВНИМАНИЕ)
!
ILM001A TESTFORT
purge (Действие подкоманды LIST M прекращается)
ILM001A TESTFORT
go (Подкоманда LIST L игнорируется, программа
   выполняется с установленной точки прерывания)
ILM400I SUBCOMMAND FOR BREAKPOINT AT 250/17 IN
   MAIN TERMINATED DUE TO ATTENSION
(Информационное сообщение о том, что выполнение
   подкоманд списка прекращено по сигналу ВНИМАНИЕ)
ILM001A TESTFORT
```

7.12. УПРАВЛЕНИЕ ОТЛАДОЧНЫМИ ДЕЙСТВИЯМИ

Последовательность планируемых абонентом отладочных действий часто зависит от выполнения некоторых условий. Эти условия можно определить в подкоманде IF, которая имеет формат:

IF (условие) подкоманда

Оба операнда позиционные. Первый операнд определяет условие IF, при котором выполняется подкоманда, заданная вторым операндом. Условие может записываться в виде отношения или логического множителя. Правила записи отношения и логического множителя такие же, как в подкоманде WHEN.

Второй операнд определяет подкоманду, которая выполняется только в том случае, если условие IF принимает значение «истина». В качестве второго операнда может быть указана любая подкоманда, включая подкоманду IF, кроме подкоманд QUALIFY и HELP. После выполнения подкоманды, указанной в IF и отличной от GO и RUN, управление опять передается абоненту: на АП выводится режимное сообщение TESTFORT. В тех случаях, когда условие IF принимает значение «ложь», управление передается абоненту сразу же после проверки заданного условия.

Пример. Массив ARRAY необходимо распечатать только в том случае, если значение переменной ALPHA равно нулю. Абонент может ввести подкоманду

```
IF(ALPHA .EQ. 0.0) LIST ARRAY
```

Если к этому моменту значение ALPHA равно нулю, протокол отладки выглядит так:

```
if (alpha .eq. 0.0) list array
ILM840I ARRAY= 1,  2,  3,  4,  5
               6,  7,  8,  9, 10
ILM001A TESTFORT
```

Если значение ALPHA отлично от нуля, протокол отладки выглядит так:

```
if (alpha .eq. 0.0) list array
ILM001A TESTFORT
```

Получив управление, абонент решает продолжить выполнение программы с оператора с меткой 10, если переменная DUPL принимает значение «истина». Для этого он вводит подкоманду IF (DUPL) GO /10.

Подкоманда IF может вводиться в списке подкоманд подкоманды AT. Она позволяет планировать отладочные действия в зависимости от заданного абонентом условия. Пусть к моменту выполнения фрагмента программы, начиная с оператора с меткой 200, необходимо, чтобы переменная ALPHA имела нулевое значение. Абонент может ввести подкоманду

```
AT /200 (IF (ALPHA .EQ. 0.0) GO; SET ALPHA=0.0; GO)
```

По этой подкоманде выполнение отладочных действий перед оператором с меткой 200 происходит следующим образом:

если значение переменной ALPHA равно нулю, выполнение программы продолжается;

если значение переменной ALPHA не равно нулю, то по подкоманде SET ее значение устанавливается равным нулю, а затем выполнение программы продолжается.

Если в подкоманде IF в качестве второго операнда используется подкоманда HALT и заданное абонентом условие принимает значение «истина», управление передается абоненту. Такая подкоманда IF позволяет прервать отладочные действия, запланированные абонентом в списке подкоманд, и выполнить другие действия с АП. Подкоманда HALT не имеет операндов.

Например, по подкоманде

```
AT /30 (IF (ITEM < 0) HALT; LIST ITEM; GO) NONOTIFY
```

выполнение отладочных действий перед оператором с меткой 30 происходит следующим образом:

если значение переменной ITEM отрицательное, управление передается абоненту для ввода подкоманд отладки;

если значение переменной ITEM равно нулю или положительное, оно выводится на АП, после чего выполнение программы продолжается с точки прерывания без уведомления абонента о прохождении программы через эту точку.

7.13. УПРАВЛЕНИЕ ОБРАБОТКОЙ ОШИБОК

Ошибки, возникающие в рабочей программе, обрабатываются программами библиотеки Фортрана. Абоненту, выполняющему в СРВ диалоговую отладку, предоставляются два способа обработки ошибок: стандартный, предусмотренный программами библиотеки Фортрана, и нестандартный, который определяется абонентом. Для управления обработкой ошибок используются подкоманды ERROR и FIXUP.

Подкоманда ERROR предназначена для установки режимов обработки ошибок в отлаживаемой программе. Режимы обработки ошибок позволяют запросить вывод на АП сообщений об ошибках и сведений об их

местоположении в программе, а также выбрать стандартный или нестандартный способ обработки ошибок.

Подкоманда ERROR имеет формат:

```
{ ERROR } { номер ошибки                    } [MSG   ] [EXIT   ]
{ ER    } { номер ошибки:номер ошибки       } [NOMSG ] [NOEXIT ]
          { (список номеров ошибок)         }
```

Первый операнд является позиционным. Он указывает номера ошибок, на которые распространяется действие режимов, заданных подкомандой ERROR. Номера ошибок могут указываться в виде списка или в виде диапазона. В списке номера записываются в порядке возрастания их значений, разделяются запятыми или пробелами, а список заключается в скобки, например ERROR (251,253,272).

Ключевые операнды MSG и NOMSG управляют выводом сообщений об ошибках. По операнду MSG сообщения об ошибках выводятся на АП. По операнду NOMSG запрещается вывод сообщений об ошибках на АП.

Ключевые операнды EXIT и NOEXIT управляют выводом сообщений о местоположении ошибок в исходной программе и выбором способа обработки ошибок. По операнду EXIT при возникновении любой из указанных в подкоманде ERROR ошибок на АП выводятся сообщение о местоположении ошибки в программном модуле и режимное сообщение TESTFORT/E, запрашивающее выполнение корректирующего действия с АП. Сообщение о местоположении ошибки выглядит следующим образом:

```
ILM867I ERROR: номер ошибки IN номер строки[/метка]IN
    имя программного модуля
```

По операнду NOEXIT при возникновении любой из указанных в подкоманде ERROR ошибок производится стандартное корректирующее действие, предусмотренное средством расширенной обработки ошибок, и выполнение программы продолжается без уведомления абонента о местоположении ошибки. По умолчанию устанавливаются режимы MSG и EXIT, по которым при возникновении ошибки на АП выводятся три сообщения. Так, если в операторе с меткой 13 головного модуля MAIN делается попытка вычислить корень из отрицательной величины, на АП выводится следующая информация:

```
IH0251I SQRT NEGATIVE ARGUMENT=—0.3600000E+02
ILM867I ERROR: 251 IN 110/13 IN MAIN
ILM002A TESTFORT/E
```

Первое сообщение содержит значение переменной (—0.3600000E+02), из которого извлекается квадратный корень. Второе сообщение содержит идентификатор оператора, в котором вычисляется квадратный корень, и имя программного модуля, которому принадлежит этот оператор. Третье сообщение уведомляет абонента о необходимости выполнения корректирующего действия. Обычно, прежде чем выполнить корректирующее действие, абонент пытается выяснить причину ошибки. Например, после получения сообщения об извлечении квадратного корня из отрицательного числа абонент может распечатать (по подкоманде LIST) значения некоторых переменных и запросить (по подкоманде WHERE) сведения о передачах управления в программе. По подкомандам отладки режимное сообщение TESTFORT/E выводится до тех пор, пока абонент не укажет корректирующее действие. Для выполнения стандартного корректирующего действия следует ввести подкоманду GO или RUN без операнда. Применительно к описываемому примеру протокол отладки будет выглядеть следующим образом:

```
IH0251I SQRT NEGATIVE ARGUMENT=—0.3600000E+02
ILM867I ERROR: 251 IN 110/13 IN MAIN
ILM002A TESTFORT/E
list a1, b1, c1
ILM840I A1=          8
ILM840I B1=         16
ILM840I C1=          0
ILM002A TESTFORT/E
go
STANDARD FIXUP TAKEN , EXECUTION CONTINUING
ILM300I AT: 120/20 IN MAIN
ILM001A TESTFORT
```

Получив уведомление об ошибке, абонент распечатывает значения коэффициентов уравнения, а затем вводит подкоманду GO. По подкоманде GO производится стандартное корректирующее действие, о чем Диалоговый отладчик уведомляет абонента выводом специального сообщения на АП, и выполнение программы продолжается до следующей точки прерывания. Напомним, что стандартное корректирующее действие заключается в том, что квадратный корень вычисляется из абсолютного значения аргумента.

В тех случаях, когда абоненту требуется выполнить корректирующее действие, отличное от стандартного, он должен использовать подкоманду FIXUP. Подкоман-

да FIXUP позволяет скорректировать значения аргументов и продолжить выполнение программы.

Подкоманда FIXUP имеет формат:

{ FIXUP / F } [ARG1 (значение)] [ARG2 (значение)]

Операнды содержат значения, присваиваемые соответственно первому и второму аргументам функции. В качестве операнда можно указать константу, имя переменной или имя элемента массива. Если один из аргументов функции не изменяется, его можно опустить. После ввода подкоманды FIXUP функция вычисляется для новых значений аргументов. При последующих вычислениях функции эти значения не сохраняются.

В рассматриваемом примере для вычисления квадратного корня из аргумента, равного нулю, абонент должен ввести подкоманду FIXUP ARG1 (0.0). В этом случае действия по корректировке выглядят следующим образом:

```
fixup arg1 (0.0)
USER FIXUP TAKEN, EXECUTION CONTINUING
ILM300I AT: 120/20 IN MAIN
ILM001A TESTFORT
```

Абонент получает уведомление о выполнении нестандартного корректирующего действия, и счет по программе продолжается до следующей точки прерывания.

Как правило, установка режимов обработки ошибок по подкоманде ERROR производится перед началом отладки программы до передачи ей управления. Однако, если у абонента возникает необходимость изменить установленные режимы в процессе отладки, он может ввести подкоманду ERROR в любой точке прерывания.

Пример. Иллюстрируются возможности абонента для управления обработкой ошибок.

Шаг 1. Абонент начинает отладку программы, состоящую из трех объектных модулей MAIN, SUB1 и SUB2, содержащихся в наборе данных PETROV.SUB12.OBJ. Идентификатор абонента PETROV.

```
testfort sub12 nosource noprint
ILM001A TESTFORT
```

Для ошибок, обнаруживаемых программами библиотеки Фортрана, по умолчанию устанавливаются режимы MSG и EXIT.

Шаг 2. Абонент предполагает, что во время выполнения программы могут возникнуть программные прерывания по исчезновению и переполнению порядка. Чтобы в этих случаях выполнение программы не останавливалось, абонент планирует стандартное

корректирующее действие с продолжением выполнения программы без уведомления об ошибке.

```
error(207,208) nomsg noexit
ILM001A TESTFORT
```

Указанные в подкоманде ERROR режимы оказывают действие только на обработку ошибок с номерами 207 и 208. Для всех остальных ошибок по-прежнему действуют режимы MSG и EXIT.

Шаг 3. Абонент планирует точки прерывания и начинает выполнение программы.

```
at /10
ILM001A TESTFORT
.   .   .
qualify sub2
ILM001A TESTFORT
at /25
ILM001A TESTFORT
go
```

Во время выполнения модуля SUB1 в операторе с номером строки 110 делается попытка вывести на печатающее устройство по формату запись, размер которой превышает максимально допустимый. Так как для ошибки с номером 212 действуют режимы MSG и EXIT, на АП выводится следующая информация:

```
IHO212I IBCOM — FORMATTED I/O,
        END OF RECORD ON UNIT 6
ILM867I ERROR: 212 IN 110 IN SUB1
ILM002A TESTFORT/E
```

Шаг 4. Абонент выполняет стандартное корректирующее действие, которое заключается в том, что формируется следующая запись, и выполнение программы продолжается.

```
go
STANDARD FIXUP TAKEN , EXECUTION CONTINUING
ILM300I AT: /10 IN MAIN
ILM001A TESTFORT
```

Шаг 5. Получив управление в запланированной точке прерывания, абонент решает, что диагностическая информация об ошибке с номером 212 ему больше не потребуется. Для этой ошибки он задает режимы, определяющие стандартное корректирующее действие и отменяющие уведомление об ошибке и приостановку программы.

```
error 212 nomsg noexit
ILM001A TESTFORT
```

Шаг 6. Абонент продолжает выполнение программы.

```
go
. . .
```

Абонент получает диагностическую информацию об ошибке с номером 241 (делается попытка возвести нулевое основание в целую отрицательную степень).

```
IHO241I FIXPI INTEGER BASE=0,INTEGER
        EXPONENT=—36,LE 0
ILM867I ERROR: 241 IN 120 IN SUB2
ILM002A TESTFORT/E
```

Шаг 7. С помощью подкоманды FIXUP абонент задает такие аргументы, чтобы результат возведения в степень стал равным единице, и продолжает выполнение программы. Программа приостанавливается в следующей точке прерывания.

```
fixup arg1(1) arg2(0)
USER FIXUP TAKEN , EXECUTION CONTINUING
ILM300I AT: 160/25 IN SUB2
ILM001A TESTFORT
```

7.14. ЗАВЕРШЕНИЕ СЕАНСА ОТЛАДКИ

Закончить сеанс отладки можно одним из двух способов в зависимости от состояния отлаживаемой программы. Если выполнение программы приостановлено в точке прерывания, т. е. абоненту разрешен ввод подкоманд, достаточно ввести подкоманду END. Если нужно прекратить сеанс отладки, когда происходит выполнение отлаживаемой программы, необходимо дважды запросить прерывание по сигналу ВНИМАНИЕ. В том и другом случае выполнение программы прекращается и устанавливается режим ввода команд СРВ. Причем в случае использования подкоманды END освобождается вся основная память, занятая Диалоговым отладчиком.

Пример. Пусть программа приостановлена в точке прерывания. Абонент завершает сеанс отладки по подкоманде END.

```
ILM300I AT: 160/25 IN SUB2
ILM001A TESTFORT
end
READY
```

7.15. ПРЕРЫВАНИЕ ПО СИГНАЛУ ВНИМАНИЕ

Во время сеанса отладки прерывание по сигналу ВНИМАНИЕ используется для установления связи с АП. Необходимость в таком действии может возникнуть в следующих случаях:

а) при вводе команды TESTFORT или любой ее подкоманды абонент допустил ошибку и испытывает затруднения в ее исправлении или абоненту требуется прекратить обработку введенной команды или подкоманды;

б) производится вывод отладочной информации на АП по одной из подкоманд LIST, LISTBRKS, LISTFREQ или WHERE. Информация, необходимая абоненту, уже получена, и требуется прекратить дальнейший вывод на АП;

в) необходимо прекратить выполнение программы, например, в случае ее зацикливания.

Прерывание по сигналу ВНИМАНИЕ позволяет прервать обработку команды TESTFORT, любой ее подкоманды, а также выполнение отлаживаемой программы.

Если абонент запросил прерывание в случае а), обработка команды или подкоманды приостанавливается и на АП выводится соответственно режимное сообщение READY или TESTFORT. Абонент может либо продолжить обработку команды или подкоманды, либо прекратить ее. Для продолжения обработки достаточно ввести пустую строку. Паузу до момента ввода пустой строки абонент может использовать для анализа результатов своей работы. Для прекращения обработки команды или подкоманды необходимо ввести соответственно любую команду СРВ или подкоманду отладки.

Если абонент запросил прерывание в случае б), то его действия определяются правилами, изложенными в 7.11.

Если абонент запросил прерывание в случае в), выводится режимное сообщение TESTFORT и выполнение программы приостанавливается. Не исключено, что при попытке приостановить выполнение программы будет прервана обработка списка подкоманд. Чтобы исключить возможность возникновения непредвиденных ситуаций, после прерывания по сигналу ВНИМАНИЕ рекомендуется прежде всего установить временную точку прерывания (по подкомандам NEXT и GO) и тем самым завершить обработку прерванной подкоманды или оператора. Если прерывается подкоманда, поступившая из списка подкоманд, она завершается, выполняются последующие подкоманды списка и оператор, к которому относится этот список. Если прерывание по сигналу ВНИМАНИЕ абонент использует в качестве паузы для анализа результатов отладки, продолжить выполнение программы он может вводом пустой строки или подкоманды GO без операнда.

Пример. Иллюстрируется использование прерывания по сигналу ВНИМАНИЕ при отладке программы.

Шаг 1. Для АП типа ЕС-7906 абонент определяет возможность прерывания по сигналу ВНИМАНИЕ через каждые 60 секунд при вводе символа «%».

```
terminal input(%) seconds(60)
READY
```

Шаг 2. Абонент, работающий с идентификатором IVANOV, приступает к отладке программы, которая хранится в виде объектных модулей в наборах данных IVANOV. SUB1. OBJ и IVANOV. SUB2. OBJ. Он предполагает использовать тексты исходных модулей, которые содержатся в наборах данных IVANOV. SUB1. FORT и IVANOV. SUB2. FORT, и набор даных печати IVANOV. SYS1. TESTFORT. LIST.

```
testfort (sub1,sub2) source print
***(Разрешение на запрос прерывания по сигналу ВНИМАНИЕ)
```

Шаг 3. До того как закончилась обработка команды TESTFORT, абонент обнаруживает, что не распределил набор данных с номером 8 для рабочей программы. Он запрашивает прерывание по сигналу ВНИМАНИЕ для того, чтобы выполнить распределение набора данных. После этого возобновляет отладку.

```
%
!
READY
allocate file(ft08f001) dataset(*)
READY
testfort (subl,subl2) source print
***
(Ввод пустой строки)
ILM001A TESTFORT
```

Шаг 4. Абонент планирует точки прерывания и начинает выполнение программы. Так как во время выполнения произошло зацикливание, абонент запрашивает прерывание по сигналу ВНИМАНИЕ, чтобы установить причину зацикливания.

```
at /17
ILM001A TESTFORT
  .   .   .
go
***
  .   .   .
***
%
!
ILM001A TESTFORT
```

Шаг 5. Абонент организует временную точку прерывания.

```
next
ILM001A TESTFORT
go
ILM301I NEXT: 450 IN SUB2
ILM001A TESTFORT
```

Шаг 6. Определив, что зацикливание произошло в подпрограмме SUB2, абонент запрашивает вывод сведений о передачах управления в программе к моменту организации временной точки прерывания.

```
where flow
ILM995I WHERE: 450 IN SUB2
ILM997I TO: 420 IN SUB2
ILM996I FROM:  250 IN SUB1
ILM997I TO:  220 IN SUB1
ILM996I FROM: 190 IN MAIN
ILM001A TESTFORT
```

Шаг 7. Проанализировав причину зацикливания, абонент возобновляет выполнение подпрограммы SUB2 с оператора с меткой 30.

```
go /30
```

7.16. СООБЩЕНИЯ ДИАЛОГОВОГО ОТЛАДЧИКА

При вводе команды TESTFORT и ее подкоманд абонент может допустить ошибки. Диалоговый отладчик обнаруживает ошибки и выдает о них информационные сообщения. Если ошибка не серьезная, вслед за информационным сообщением следует подсказывающее сообщение REENTER, предоставляющее абоненту возможность исправить ошибку. Если ошибка серьезная, обработка команды или подкоманды прекращается и устанавливается соответственно режим ввода команд или подкоманд, позволяющий ввести правильный текст команды или подкоманды.

Так как сообщения об ошибках, допущенных в команде TESTFORT и ее подкомандах, выдаются не только Диалоговым отладчиком, но и другими компонентами СРВ, то они могут иметь префиксы ILM и IKJ. Сообщения Диалогового отладчика, имеющие префикс ILM, приведены в приложении 1.

7.16.1. Исправление ошибок в команде TESTFORT

В случае несерьезной ошибки, допущенной при вводе команды TESTFORT, выдаются информационное и подсказывающее сообщения, позволяющие исправить ошибку. Действия абонента зависят от характера ошибки. Например, при вводе неправильного имени в ответ на сообщение REENTER достаточно ввести правильное имя, чтобы обработка команды была продолжена. Если ошибки связаны с распределением наборов данных

(например, имя набора данных отсутствует в каталоге или набор данных отсутствует на томе), действия абонента зависят от того, какой это набор данных. В случае ошибки во время распределения набора данных с исходной программой абонент может ввести пустую строку и тем самым продолжить обработку команды, игнорируя его распределение. После этого набор данных становится недоступным абоненту в этом сеансе отладки. В случае ошибки во время распределения набора данных печати абонент может запросить прерывание по сигналу ВНИМАНИЕ, чтобы прекратить обработку введенной команды TESTFORT, а затем ввести команду TESTFORT с операндом NOPRINT. Вся отладочная информация в таком сеансе будет выводиться на АП. Если набор данных с исходной программой или набор данных печати требуется в сеансе отладки, абонент может запросить прерывание по сигналу ВНИМАНИЕ, чтобы вернуться в режим ввода команд СРВ. Средствами СРВ и операционной системы следует проанализировать причину ошибки, устранить ее, а затем ввести команду TESTFORT повторно.

Наборы данных с отлаживаемыми модулями и личные библиотеки должны распределяться всегда. Поэтому ошибки, обнаруженные при их распределении, являются серьезными и после выдачи информационного сообщения, указывающего причину ошибки, обработка команды TESTFORT прекращается.

Пример. Идентификатор абонента IVANOV.

testfort sortab source(square(ro)) noprint

ILM102I DATA SET SQUARE.FORT NOT ALLOCATED,
DATA SET NOT ON VOLUME+

(Информационное сообщение, указывающее, что каталогизированный набор данных с именем IVANOV.SQUARE.FORT отсутствует на томе. Символ «+» указывает на наличие сообщения второго уровня)

IKJ56703A REENTER —

(Подсказывающее сообщение о корректирующем действии)

? (Запрос на вывод сообщения второго уровня, уточняющего информационное сообщение)

ILM102I CATALOG INFORMATION INCORRECT

(Сообщение второго уровня, указывающее, что информация каталога неверная)

IKJ56703A REENTER —

(Подсказывающее сообщение о корректирующем действии)

Ввод пустой строки (Абонент решает не использовать набор данных с исходной программой в сеансе отладки)

ILM001A TESTFORT

7.16.2. Исправление ошибок в подкомандах отладки

В случае ошибки, допущенной в подкоманде отладки, выдается либо подсказывающее сообщение REENTER, либо режимное сообщение TESTFORT. В ответ на подсказывающее сообщение необходимо ввести только ту часть подкоманды, в которой допущена ошибка. Например, если неправильно заданы границы диапазона операторов или указан оператор, отсутствующий в программном модуле, достаточно ввести правильный диапазон или правильный идентификатор оператора. Если абонент введет всю подкоманду, он опять получит сообщение об ошибке. В тех случаях, когда абонент испытывает затруднения, связанные с исправлением ошибки, он может запросить прерывание по сигналу ВНИМАНИЕ, чтобы вернуться в режим ввода подкоманд и ввести всю подкоманду повторно, устранив в ней допущенные ошибки.

Пример 1.

```
go /100
ILM800I GO STATEMENT ID NOT IN CURRENT PROGRAM.
    GO IGNORED.
(Информационное сообщение, указывающее, что оператора с меткой
100 нет в программном модуле)
IKJ56703A REENTER —
(Подсказывающее сообщение о корректирующем действии)
/120 (Абонент вводит другую метку)
ILM001A TESTFORT
```

Пример 2.

```
list m(25)
ILM240I A SUBSCRIPT EXCEEDS ITS DIMENSION IN M(25)
(Информационное сообщение, указывающее, что в элементе массива
задан индекс, превосходящий верхнюю границу измерения массива)
ILM001A TESTFORT
list m(20)
ILM840I M(20) = 245
ILM001A TESTFORT
```

7.17. ПРИМЕР СЕАНСА ОТЛАДКИ ПРОГРАММЫ

Производится отладка программы УРАВНЕНИЯ. В программе вычисляются корни квадратных уравнений вида: $Ax^2+Bx+C=0$. В качестве устройств ввода и вывода данных используется АП. По оператору READ из набора данных с номером 5 вводятся коэффициенты для двух уравнений. Признаком конца данных является нулевое значение первого коэффициента. Результаты вычисления корней уравнения выводятся в виде

```
      COMMON  X1REAL,X1IMAG,X2REAL,X2IMAG                              00000010
      NPAGE=1                                                          00000020
    5 WRITE(6,8)NPAGE                                                  00000030
    8 FORMAT(1H1,4X,1HA,7X,1HB,7X,1HC,3X,'X1 REAL',1X,'X1 IMAG',1X,    00000040
     1'X2 REAL',1X,'X2 IMAG',2X,'PAGE= ',I4//)                         00000050
      NPAGE=NPAGE+1                                                    00000060
      LINES=0                                                          00000070
   14 READ(5,15)A1,B1,C1,A2,B2,C2                                      00000080
   15 FORMAT(6F6.0)                                                    00000090
      IF (A1) 16,17,16                                                 00000100
   17 STOP                                                             00000110
   16 CALL SOLVE(A1,B1,C1)                                             00000120
      CALL OUTPUT(A1,B1,C1)                                            00000130
      IF (A2)  13,17,13                                                00000140
   13 CALL SOLVE(A2,B2,C2)                                             00000150
      CALL OUTPUT(A2,B2,C2)                                            00000160
      LINES=LINES+2                                                    00000170
      IF (LINES-12) 14,5,5                                             00000180
      END                                                              00000190
```

Рис. 3. Текст головного модуля MAIN

```
      SUBROUTINE  SOLVE(A,B,C)                                         00000200
      COMMON  X1REAL,X1IMAG,X2REAL,X2IMAG                              00000210
      DISC=B**2-4.0*A*C                                                00000220
      IF  (DISC)50,60,70                                               00000230
   50 X1REAL=-B/(2.0*A)                                                00000240
      X2REAL=X1REAL                                                    00000250
      X1IMAG=SQRT(-DISC)/(2.0*A)                                       00000260
      X2IMAG=-X1IMAG                                                   00000270
      RETURN                                                           00000280
   60 X1REAL=-B/(2.0*A)                                                00000290
      X2REAL=X1REAL                                                    00000300
      X1IMAG=0.0                                                       00000310
      X2IMAG=0.0                                                       00000320
      RETURN                                                           00000330
   70 S=SQRT(DISC)                                                     00000340
      X1REAL=(-B+S)/(2.0*A)                                            00000350
      X2REAL=(-B-S)/(2.0*A)                                            00000360
      X1IMAG=0.0                                                       00000370
      X2IMAG=0.0                                                       00000380
      RETURN                                                           00000390
      END                                                              00000400
```

Рис. 4. Текст подпрограммы SOLVE

```
      SUBROUTINE OUTPUT(A,B,C)                                    00000410
      COMMON X1REAL,X1IMAG,X2REAL,X2IMAG                          00000420
      IF(X1IMAG)90,91,90                                          00000430
   91 WRITE(6,95) A,B,C,X1REAL,X2REAL                             00000440
   95 FORMAT('0',1P4E8.1,8X,1PE8.1)                               00000450
      RETURN                                                      00000460
   90 IF(X1REAL)100,101,100                                       00000470
  101 IF(X2REAL)100,102,100                                       00000480
  102 WRITE(6,103)A,B,C,X1IMAG,X2IMAG                             00000490
  103 FORMAT('0',1P3E8.1,8X,1PE8.1,8X,1PE8.1)                     00000500
      RETURN                                                      00000510
  100 WRITE(6,110)A,B,C,X1REAL,X1IMAG,X2REAL,X2IMAG               00000520
  110 FORMAT('0',1P7E8.1)                                         00000530
      RETURN                                                      00000540
      END                                                         00000550
```

Рис. 5. Текст подпрограммы OUTPUT

таблицы в набор данных с номером 6. Программа состоит из головного модуля MAIN и двух подпрограмм SOLVE и OUTPUT. В головном модуле MAIN производится ввод данных, печать заголовка таблицы и нумерация страниц. В подпрограмме SOLVE вычисляются корни уравнения. Коэффициенты уравнения являются параметрами подпрограммы. В подпрограмме OUTPUT выполняется печать строк таблицы. Каждая строка содержит значения коэффициентов и значения корней уравнения. Если действительная или мнимая часть корня нулевая, то соответствующее поле печати заполняется пробелами. Текст программы приведен на рис. 3—5. Исходный текст размещается в последовательном наборе данных с именем TERM1.SQUARE.FORT. Сеанс отладки состоит из 13 шагов.

Шаг 1. Используя команду LOGON, абонент начинает сеанс работы. В команде LOGON указываются идентификатор абонента TERM1 и имя процедуры DBGFORT.

Шаг 2. По команде PROFILE абонент устанавливает режим вывода подсказывающих сообщений (операнд PROMPT) и режим вывода идентификаторов сообщений (операнд MSGID).

Шаг 3. Для трансляции программы абонент использует команду FORTSE с операндом TEST. Результаты трансляции (объектные модули) помещаются в набор TERM1. SQUARE. OBJ.

Шаг 4. Отладку программы абонент начинает вводом команды TESTFORT. Операнд SOURCE обеспечивает доступ к набору данных TERM1. SQUARE. FORT, содержащему текст программных модулей MAIN, SOLVE и OUTPUT. Операнд NOPRINT задает вывод отладочной информации только на АП.

Шаг 5. Используя подкоманды AT, абонент планирует три точки прерывания в головном модуле MAIN для выполнения отладочных действий с АП. Дополнительно в первой точке прерывания он задает вывод значения переменной NPAGE, используемой при подсчете страниц. С помощью подкоманды TRACE абонент запрашивает вывод на АП сообщений о входах и выходах в подпрограммы SOLVE и OUTPUT и о передачах управления в программных модулях.

Используя подкоманды QUALIFY и AT, абонент планирует четыре точки прерывания в подпрограмме SOLVE. Первая точка прерывания устанавливается для вывода значения дискриминанта. Абонент планирует точку прерывания так, чтобы не получать управление и уведомление о прохождении программы через эту точку. Три другие точки прерывания устанавливаются, чтобы распечатать значения корней уравнения без передачи управления абоненту, но с уведомлением его о прохождении программы через эти точки.

Шаг 6. С помощью подкоманды GO абонент начинает выполнение программы. По оператору WRITE на АП выводится заголовок таблицы. В соответствии с действиями, запланированными в точке прерывания, установленной в строке 80, на АП выводится значение переменной NPAGE и выполнение программы приостанавливается. Управление передается абоненту.

Шаг 7. Для присваивания значений коэффициентам двух уравнений абонент использует подкоманды SET.

Шаг 8. С помощью подкоманды GO абонент возобновляет выполнение программы с оператора в строке с номером 100, чтобы обойти ввод значений коэффициентов по формату. Вычисляются корни уравнения. На АП выводится отладочная информация, запланированная абонентом на пятом шаге (сообщения о входах и выходах в подпрограммы SOLVE и OUTPUT, переходах внутри этих подпрограмм, значение дискриминанта, значения действительной и мнимой частей корней уравнения), а также строка таблицы по оператору WRITE. Выполнение программы приостанавливается в головном модуле (оператор в строке 180), и управление передается абоненту.

Шаг 9. Убедившись по результатам счета, что проверяемые части алгоритма выполняются верно, абонент продолжает отладку. С помощью подкоманды LIST распечатывает значение переменной LINES, используемой для подсчета строк на странице. По подкоманде OFF удаляет первую точку прерывания в головном модуле (оператор в строке 80).

Шаг 10. По подкоманде GO абонент возобновляет выполнение программы. Ввод значений коэффициентов теперь выполняется по оператору READ. При вводе первого коэффициента абонент допустил ошибку: вместо значения 1 ввел символ «l». В связи с действующими по умолчанию режимами обработки ошибок MSG и EXIT на АП выводится сообщение об ошибке, и управление передается абоненту.

Шаг 11. Абонент решает выполнить нестандартное корректирующее действие. Так как для ошибки с номером 215 нельзя использовать подкоманду FIXUP, абонент вводит подкоманды NEXT и GO. Выполняется стандартное корректирующее действие и устанавливается временная точка прерывания. Абонент исправляет ошибку (подкоманда SET) и продолжает выполнение программы (подкоманда GO). На АП выводится отладочная информация и строка

таблицы. Выполнение программы приостанавливается в точке прерывания (оператор с меткой 17).

Шаг 12. Абонент анализирует отладочную информацию и убеждается, что проверяемые части алгоритма программы работают верно. Дальнейшие вычисления он планирует без отладочных действий, поэтому возобновляет выполнение программы по подкоманде RUN с оператора в строке 30. Последующие вычисления производятся для двух уравнений с коэффициентами 1, 0, 10 и 1, 8, 16. При очередном запросе на ввод коэффициентов абонент вводит признак конца данных — значение 0, чтобы закончить выполнение.

Шаг 13. Получив на АП сообщение о нормальном окончании программы, абонент решает проверить, все ли операторы подпрограммы SOLVE выполнялись при отладке. Установив программный уточнитель, равный имени SOLVE, по подкоманде LISTFREQ он запрашивает сведения об операторах, которые не получали управление. Анализируя вывод подкоманды LISTFREQ, абонент убеждается, что не работающих операторов в программе нет. По подкоманде END абонент заканчивает отладку, а по команде LOGOFF завершает сеанс работы.

Протокол сеанса работы имеет следующий вид:

```
logon term1 proc(dbgfort)                                    Шаг 1
IKJ56455I TERM1 LOGON IN PROGRESS  AT 12:59:53
    ON JANUARY 10,1981
IKJ56951I NO BROADCAST MESSAGES
READY
profile prompt msgid                                         Шаг 2
READY
fortse square test load noprint noterm                       Шаг 3
READY
testfort square noprint source                               Шаг 4
ILM001A TESTFORT
at 80 (list npage)                                           Шаг 5
ILM001A TESTFORT
at (/17 180)
ILM001A TESTFORT
trace stmt
ILM001A TESTFORT
qualify solve
ILM001A TESTFORT
at 230 (list disc;go) nonotify
ILM001A TESTFORT
at (280 330 390) (list (x1real x2real x1imag x2imag); go)
ILM001A TESTFORT
go                                                           Шаг 6
A   B   C   X1  REAL   X1  IMAG   X2  REAL
    X2  IMAG   PAGE = 1
ILM300I AT: 80/14 IN MAIN
ILM840I NPAGE=       2
ILM00IA TESTFORT
set a1=1                                                     Шаг 7
ILM001A TESTFORT
set b1=2
ILM001A TESTFORT
```

```
set c1=10
ILM001A TESTFORT
set a2=1
ILM001A TESTFORT
set b2=8
ILM001A TESTFORT
set c2=16
ILM001A TESTFORT
go 100                                                    Шаг 8
ILM304I TRACE: FROM 100 TO 120/16
ILM302I TRACE: SOLVE ENTERED
ILM840I DISC=—36.0000000
ILM300I AT: 280 IN SOLVE
ILM840I X1REAL=—1.00000000
ILM840I X2REAL=—1.00000000
ILM840I X1IMAG= 3.00000000
ILM840I X2IMAG= —3.00000000
ILM305I TRACE: RETURN FROM SOLVE
ILM302I TRACE: OUTPUT ENTERED
ILM304I TRACE: FROM 430 TO 470/90
ILM304I TRACE: FROM 470/90 TO 520/100
1.0E+00 2.0E+00 1.0E+01—1.0E+00 3.0E+00—1.0E+00—3.0E+00
ILM305I TRACE: RETURN FROM OUTPUT
ILM302I TRACE: SOLVE ENTERED
ILM840I DISC=      .0
ILM304I TRACE: FROM 230 TO 290/60
ILM300I AT: 330 IN SOLVE
ILM840I X1REAL= —4.00000000
ILM840I X2REAL= —4.00000000
ILM840I X1IMAG=      .0
ILM840I X2IMAG=      .0
ILM305I TRACE: RETURN FROM SOLVE
ILM302I TRACE: OUTPUT ENTERED
1.0E+00 8.0E+00 1.6E+01—4.0E+00      —4.0E+00
ILM305I TRACE: RETURN FROM OUTPUT
ILM300I AT: 180 IN MAIN
ILM001A TESTFORT
list lines                                                Шаг 9
ILM840I LINES=            2
ILM001A TESTFORT
off 80
ILM001A TESTFORT
go                                                        Шаг 10
ILM304I TRACE: FROM 180 TO 80/14
   !      5       4       0
IH0215I CONVERT—ILLEGAL DECIMAL CHARACTER !
   !      5       4       0
ILM867I ERROR: 215 IN 80/14 IN MAIN
ILM002A TESTFORT/E
next                                                      Шаг 11
ILM002A TESTFORT/E
go
STANDARD FIXUP TAKEN, EXECUTION CONTINUING
ILM301I NEXT: 100 IN MAIN
ILM001A TESTFORT
```

```
set a1= 1
ILM001A TESTFORT
go
ILM304I TRACE: FROM 100 TO 120/16
ILM302I TRACE: SOLVE ENTERED
ILM840I DISC= 9.00000000
ILM304I TRACE: FROM 230 TO 340/70
ILM300I AT: 390 IN SOLVE
ILM840I X1REAL= —1.00000000
ILM840I X2REAL= —4.00000000
ILM840I X1IMAG=  .0
ILM840I X2IMAG=  .0
ILM305I TRACE: RETURN FROM SOLVE
ILM302I TRACE: OUTPUT ENTERED
1.0E+00 5.0E+00 4.0E+00—1.0E+00        —4.0E+00
ILM305I TRACE: RETURN FROM OUTPUT
ILM304I TRACE: FROM 140 TO 110/17
ILM300I AT: 110/17 IN MAIN
ILM001A TESTFORT
run 30                                              Шаг 12
A    B    C    X1 REAL X1 IMAG X2 REAL X2  IMAG PAGE=2
1    0     10     1     8     16
1.0E+00  0.0      1.0E+01 3.2E+00      —3.2E+00
1.0E+00  8.0E+00 1.6E+01—4.0E+00  —4.0E+00
0
SUMMARY OF ERRORS FOR THIS JOB
          ERROR NUMBER NUMBER OF ERRORS
          215                  1
ILM306I PROGRAM UNDER TESTFORT HAS
    TERMINATED NORMALLY
ILM001A TESTFORT
qualify solve                                       Шаг 13
ILM001A TESTFORT
listfreq zerofreq
ILM940I ZERO—FREQUENCY STATEMENTS IN SOLVE
ILM943I NONE
ILM001A TESTFORT
end
READY
logoff
IKJ56470I TERM1 LOGGED OFF TSO AT 13:15:05
          ON JANUARY 10, 1981
```

Глава 8

ПОДГОТОВКА СИСТЕМЫ ПРОГРАММИРОВАНИЯ ФОРТРАН К РАБОТЕ

8.1. ТРЕБОВАНИЯ К ОПЕРАЦИОННОЙ СИСТЕМЕ

Система программирования Фортран для СРВ оформлена в виде пакета прикладных программ и поставляется пользователям на отдельной дистрибутивной магнитной ленте вместе с собственными средствами включения в операционную систему. Процедура включения, называемая также генерацией, организована таким образом, что позволяет создавать систему Фортран любой конфигурации. Системному программисту, ответственному за создание системы, предоставляется возможность на этапе генерации планировать состав системы Фортран, а также режимы работы отдельных ее компонентов.

На дистрибутивной ленте система Фортран разбита на следующие части, являющиеся единицами включения: транслятор Фортран SE, транслятор Фортран CC и программа преобразования, посредник транслятора Фортран SE, посредники транслятора Фортран CC и программы преобразования, библиотека программ Фортрана, Диалоговый отладчик. Из этих частей можно собирать систему Фортран любой конфигурации. При определении состава системы Фортран для конкретного применения на вычислительном центре следует учитывать такие факторы, как режимы работы операционной системы (пакетный или СРВ), потребность в одном или двух трансляторах, возможность ведения диалоговой отладки программ. Независимо от режима работы операционной системы необходимо обеспечить включение библиотеки программ Фортрана, которая поддерживает оба транслятора и Диалоговый отладчик. Наличие посредников и Диалогового отладчика обязательно в тех случаях, когда трансляцию и отладку программы планируется выполнять в СРВ.

Для задания режимов выполнения трансляторов и библиотеки программ система Фортран содержит набор макрокоманд генерации. Режимы трансляции, указанные в макрокомандах, становятся режимами, действующими по умолчанию при вызове трансляторов.

Подготовка к включению системы Фортран состоит из нескольких шагов, на которых системный программист должен: определить покомпонентный состав системы Фортран; описать библиотеки, в которые будет производиться включение системы Фортран (допускается использование личных и системных библиотек); указать режимы работы трансляторов и библиотеки программ Фортрана.

Свои требования к системе Фортран программист оформляет в виде отдельного задания. Выполнение этого задания составляет первый этап включения. На втором этапе производится выполнение заданий, размещенных на дистрибутивной магнитной ленте вместе с компонентами системы Фортран. После завершения второго этапа включения пополняются системные библиотеки или создается до шести новых библиотек, содержащих: загрузочные модули трансляторов и Диалогового отладчика; каталогизированные процедуры для вызова трансляторов; программы библиотеки Фортрана; посредники системы Фортран; справочную информацию о командах СРВ; управляющий модуль Диалогового отладчика.

В зависимости от выбранной конфигурации системы Фортран могут создаваться все или только некоторые из указанных библиотек.

При включении компонентов системы программирования Фортран в ОС ЕС к операционной системе не предъявляются особые требования. Обычно стандартная конфигурация операционной системы вполне достаточна для того, чтобы выполнить процедуру включения. Необходимо обеспечить только наличие 176 Кбайт основной памяти, программ обслуживания IEBUPDTE, IEHPROGM, IEBGENER, транслятора с языка Ассемблера и Редактора. Для размещения всех компонентов системы Фортран на дисках требуется память в объеме 150 дорожек (ЕС-5061) или 300 дорожек (ЕС-5050). Сама система программирования Фортран функционирует в операционной системе ОС ЕС, начиная с издания 6.1, включающего систему разделения времени MVT издания 1.2 или SVS издания 1.0. Для использования всех возможностей системы Фортран требуется от 142 до 166 Кбайт основной памяти и наличие в операционной системе Загрузчика и обработчика команды СРВ LOADGO. Эти программы вызываются соответственно

транслятором Фортран CC и Диалоговым отладчиком для выполнения действий, связанных с редактированием программы.

8.2. ТРЕБОВАНИЯ К ПРОЦЕДУРЕ LOGON

Чтобы начать эксплуатацию системы Фортран в СРВ, необходимо выполнить некоторые подготовительные действия. Одним из таких действий является создание процедуры LOGON, соответствующей выбранной конфигурации системы Фортран. Требования к процедуре LOGON определяются системным программистом вместе с абонентами. При этом учитываются такие факторы, как программные компоненты системы Фортран, с которыми предполагают работать абоненты, и используемые ими наборы данных. Обычно достаточно иметь одну универсальную процедуру LOGON для работы в СРВ со всеми компонентами системы Фортран.

В общем случае процедура LOGON состоит из оператора EXEC, инициирующего выполнение задания в СРВ, и операторов DD, описывающих используемые в задании наборы данных. Часть наборов данных распределяется постоянно, остальные — динамически. Постоянно рекомендуется распределить наборы данных, необходимые на протяжении всего сеанса работы. К таким наборам данных относятся личные библиотеки, содержащие программные компоненты системы Фортран, а также наборы данных, стандартно используемые обработчиком команды EDIT, трансляторами, Диалоговым отладчиком и рабочей программой. Наборы данных, распределяемые постоянно, описываются с помощью операторов DD, имеющих стандартные имена. Имена операторов DD и соответствующие им наборы данных приведены ниже.

SYSEDIT, SYSUT1 — временные наборы данных, используемые командой EDIT при создании и корректировке наборов данных в СРВ.

STEPLIB — личная библиотека, содержащая загрузочные модули компонентов системы Фортран. С этим набором следует объединить набор данных, содержащий загрузочные модули посредников, если они не включены в набор данных SYS1.CMDLIB.

SYSHELP — личная библиотека, содержащая справочную информацию о командах и подкомандах систе-

мы Фортран. Эта библиотека должна быть объединена с библиотекой SYS1.HELP.

SYSLIB — личная библиотека, содержащая управляющий модуль Диалогового отладчика. С этим набором следует объединить библиотеку программ Фортрана SYS1.FORTLIB.

SYSABEND или SYSUDUMP — наборы данных для вывода дампов при ненормальном завершении программы.

SYSTERM — набор данных для вывода на АП диагностических сообщений транслятора Фортран SE, Загрузчика и Редактора.

CNVOUT — набор данных для вывода на АП диагностических сообщений программы преобразования.

FT05F001, FT06F001 — наборы данных рабочей программы, используемые обычно для ввода и вывода информации на АП.

Кроме постоянно распределенных наборов данных, в процедуре LOGON необходимо зарезервировать достаточное количество операторов DD с операндом DYNAM для динамического распределения наборов данных во время выполнения системы Фортран. Ниже приведены требования компонентов системы Фортран к наборам данных. Количество динамически распределяемых наборов данных указано с учетом того, что часть наборов данных распределена постоянно.

Транслятор Фортран SE — набор данных с исходной программой и наборы данных для вывода объектных модулей и распечаток трансляции.

Транслятор Фортран CC — набор данных с исходной программой и набор данных для вывода объектных модулей.

Программа преобразования — наборы данных с исходной и преобразованной программами.

Диалоговый отладчик — наборы данных с объектными и загрузочными модулями отлаживаемой программы; наборы данных с исходной программой, но не более 6; личные библиотеки с загрузочными модулями отлаживаемой программы; набор данных печати.

Рабочая программа — наборы данных, обращение к которым производится по номерам, отличным от 5 и 6.

Общее количество наборов данных, распределяемых динамически, должно обеспечивать выполнение любого компонента системы Фортран и любой рабочей прог-

раммы. Пример процедуры LOGON (имя процедуры FORTRAN), обеспечивающей выполнение системы Фортран, включенной в личные библиотеки, приведен ниже.

Пример.

```
//FORTRAN   EXEC PGM=IKJEFT01,PARM='TERM INPUT(%)
//                ',ROLL=(NO,NO)
//SYSEDIT   DD   DSN=&EDIT,UNIT=SYSDA,
//                SPACE=(1688,(50,20))
//SYSUT1    DD   DSN=&SYSUT1,UNIT=SYSDA,
//                SPACE=(CYL,(10,10))
//STEPLIB   DD   DSN=LINKLIBF,DISP=SHR
//          DD   DSN=CMDLIBF,DISP=SHR
//SYSHELP   DD   DSN=HELPF,DISP=SHR
//          DD   DSN=SYS1.HELP,DISP=SHR
//SYSLIB    DD   DSN=SYS1.TFORTLIB,DISP=SHR
//          DD   DSN=SYS1.FORTLIB,DISP=SHR
//SYSUDUMP  DD   SYSOUT=A,SPACE=(TRK,(100,50))
//SYSTERM   DD   TERM=TS
//CNVOUT    DD   TERM=TS
//FT05F001  DD   TERM=TS
//FT06F001  DD   TERM=TS
//DD1       DD   DYNAM
//DD2       DD   DYNAM
  .         .    .    .
//DD10      DD   DYNAM
```

В данном примере загрузочные модули трансляторов, программы преобразования и Диалогового отладчика содержатся в наборе данных с именем LINKLIBF, посредники помещены в набор данных с именем CMDLIBF, справочная информация о командах и подкомандах — в набор данных с именем HELPF. Программы библиотеки Фортрана содержатся в наборе данных SYS1.FORTLIB, а управляющий модуль Диалогового отладчика — в наборе данных SYS1.TFORTLIB. Для динамического распределения предназначены 10 наборов данных.

ЛИТЕРАТУРА

1. Фортран ЕС ЭВМ / З. С. Брич, Д. В. Капилевич, С. Ю. Котик, В. И. Цагельский. М., Статистика, 1978.
2. Система математического обеспечения ЕС ЭВМ / Под общ. ред. А. М. Ларионова. М., Статистика, 1974.
3. **Бертэн Ж., Риту М., Ружие Ж.** Работа ЭВМ с разделением времени. М., Наука, 1970.
4. Программирование на языке Ассемблера ЕС ЭВМ / З. С. Брич, В. И. Воюш, Г. С. Дегтярева, Э. В. Ковалевич. М., Статистика, 1975.

ПРИЛОЖЕНИЯ

Приложение 1

СООБЩЕНИЯ КОМПОНЕНТОВ СИСТЕМЫ ФОРТРАН

Сообщения транслятора Фортран СС (префикс IGK)

100(4/16) — Произошла ошибка ввода-вывода.
192(16) — Недостаточно основной памяти.
193(16) — Превышен размер таблицы транслятора.
194(16) — Невозможно открыть набор данных SYSLIB.
195(16) — Невозможно открыть набор данных SYSIN.
196(16) — В задании отсутствует оператор DD.
197, 198(16) — Ошибка вычислительной системы.
200(16) — Недостаточно памяти либо ошибка оборудования.
401(0) — Отсутствует запятая.
402(0) — Тип константы в DATA или в объявлении типа отличен от типа переменной.
403(0) — Отсутствует END.
404(0) — Вместо STOP используется RETURN.
405(0) — В модуле-функции имени функции или имени входа не присвоено значение.
407(0) — В операторе READ или WRITE, управляемом списком, отсутствует список ввода-вывода.
410(4) — В имени более 6 символов.
411(4) — Количество параметров, указанных в обращении к программе библиотеки Фортрана, неверное.
412(4) — Переменным или массивам из блока COMMON не может быть распределена память из-за ошибок.
413(4) — Ошибка размещения переменных в EQUIVALENCE.
414(4) — В объявлении EQUIVALENCE используется элемент необъявленного массива.
415(4) — Ошибка при объявлении массива с регулируемыми размерами.
416(4) — В модуле BLOCK DATA присваивается начальное значение переменной не помеченного блока COMMON.
418(4) — Невозможно открыть набор данных SYSPUNCH.
420(4) — У оператора должна быть метка.
421(4) — В объявлении FORMAT отсутствует метка.
422(4) — Объявление содержит метку.
423(4) — Формат записей набора данных с исходной программой не соответствует формату, указанному в операторе DD.
424(4) — В объявлении DATA или в явном объявлении типа размер текстовой константы превышает размер соответствующей переменной или элемента массива или массиву присваиваются значения одной текстовой константой.
464(8) — Объявление BLOCK DATA не является первым.
465(4) — Объявление используется в логическом IF.
466(8) — В операторе PAUSE или STOP указано более 5 цифр.
467(8) — Имя списка в NAMELIST не уникальное.
468(8) — Имя списка NAMELIST ранее использовалось как имя массива.
469(8) — Отсутствует или неправильно используется символ «/».
470, 472(8) — Имя процедуры ранее использовалось как имя переменной или имя массива.

471, 473(8) — Неправильное использование объявления EXTERNAL.
474(8) — Переменная используется в двух блоках COMMON.
475(8) — Имеются незакрытые циклы DO.
476(8) — Используются неопределенные метки.
477(8) — Формальный параметр процедуры используется в COMMON, NAMELIST или DISPLAY.
478(8) — Объявление ENTRY содержится в пакете отладки или в модуле BLOCK DATA.
479(8) — Объявление ENTRY содержится внутри тела цикла.
480(8) — ENTRY используется в головном модуле.
482(8) — Отсутствует запятая.
483(8) — Цикл DO заканчивается запрещенным оператором.
484(8) — Оператор цикла используется в логическом IF.
485(8) — В операторе цикла или в списке с циклом пропущен или не на месте символ «=».
486(8) — В операторе цикла или в списке с циклом пропущена или неправильно используется запятая.
487(8) — Формат предложения не соответствует языку Фортран.
488(8) — END используется в логическом IF.
489(8) — Символ не соответствует синтаксису языка Фортран.
490(8) — Используется неправильная константа для масштабного множителя.
491(8) — Масштабный множитель по модулю превышает 128.
492(8) — Описатель масштабного множителя указан с кодом преобразования, отличным от F, E, D или G.
493(8) — В объявлении FORMAT неправильно используются скобки.
494(8) — В объявлении FORMAT для кода преобразования указан недопустимый символ.
495(8) — В объявлении FORMAT код преобразования не указан или указан неверно.
496(8) — В описателе поля формата для кода преобразования E, F или D пропущена точка.
497(8) — Количество цифр дробной части, указанное в описателе поля формата, превосходит общую длину числа.
499(8) — Значение n, w, d или r, указанное в описателе поля формата, превышает 255.
500(8) — Отсутствует или неправильно используется символ «(».
501(8) — Литерал содержит более 255 символов.
502(8) — В вычисляемом GO TO после символа «)» следует недопустимый символ.
504(8) — Отсутствует или неправильно используется символ «)».
505(8) — В операторе ASSIGN пропущены или неправильно используются символы «TO».
506(8) — Неправильно указана запятая при записи метки перехода.
507(8) — В логическом IF содержится другой логический IF.
508(8) — Неверное размещение объявления IMPLICIT.
509(8) — Объявление IMPLICIT содержит недействительный тип.
510(8) — В объявлении IMPLICIT или в явном объявлении типа используется недопустимый символ.
511(8) — Отсутствует символ «)».
512(8) — Противоречие в указании типа в явном или неявном объявлении типа.
513(8) — В предложении указана неверная метка.
514(8) — Метка уже использовалась как метка другого предложения.

515(8) — В операторе ввода-вывода используется метка не объявления FORMAT.
516(8) — Предложение не закончено, должна следовать строка продолжения.
518(8) — Предложение, на которое указывает метка перехода, не является оператором.
519(8) — Метка в операторе управления указывает на объявление FORMAT.
520(8) — В программном модуле отсутствуют правильные предложения Фортрана.
521(8) — Строка первого предложения является строкой продолжения.
522(8) — Предложение содержит более 19 строк продолжения или предложение в свободном формате состоит более чем из 1320 символов.
523, 525(8) — Предложение содержит символ, не соответствующий синтаксису языка Фортран.
524(8) — Не закончена запись литерала.
526(8) — Для элемента массива не указан индекс.
527(8) — В операторе не на месте символ «=».
528(8) — Оператор присваивания начинается не с переменной или элемента массива.
529(8) — Недействительный оператор в логическом IF.
530(8) — Объявление внутренней функции содержит не уникальные формальные параметры.
531(8) — Логической переменной присваивается значение выражения, отличного от логического.
532(8) — Переменной не логического типа присваивается значение логического выражения.
533(8) — Для заданного типа указана неверная длина.
534(8) — Границы массива определены более чем в одном объявлении.
535(8) — Переменная определена в объявлениях типов, противоречащих друг другу.
536, 537(8) — Имя процедуры ранее определено как имя массива.
538(8) — Имя процедуры используется в объявлении ENTRY или как формальный параметр.
540(8) — В объявлении DEFINE FILE опущен или неправильно используется параметр L.
541(8) — Неправильное размещение заголовка процедуры в программном модуле.
542(8) — Отсутствует или неправильно используется символ «/».
543(8) — Неправильно используется символ «*».
544, 545, 546(8) — Формальный параметр используется в объявлении EQUIVALENCE или COMMON.
547(8) — Используется элемент необъявленного массива.
548(8) — Для элемента массива указан неправильный индекс.
549(8) — Отрицательное число предшествует символу «*» в объявлении DATA.
550(8) — Количество переменных, которым присваиваются значения, не согласуется с количеством констант.
551(8) — В объявлении DATA переменной не логического типа присваивается начальное значение логического типа.
552(8) — Шестнадцатеричная или логическая константа указана со знаком.
553(8) — Для элемента массива указан индекс не целого типа.

554(8) — Формальному параметру присваивается начальное значение в объявлении DATA или в объявлении типа.
555(8) — Объявление DATA или объявление типа содержится в пакете отладки.
556(8) — Ошибочная инициализация объектов блока COMMON.
557(8) — Переменная определена в объявлениях типов, противоречащих друг другу.
558(8) — Номер записи в операторе ввода-вывода прямого доступа не является выражением целого типа.
559(8) — В операторе ввода-вывода вместо метки объявления FORMAT или имени массива используется имя переменной.
563(8) — В операторе FIND после номера набора данных отсутствует апостроф.
564(8) — Противоречие в использовании формального параметра.
565(8) — Количество измерений в элементе массива больше 7.
567(8) — Имя списка в объявлении NAMELIST не уникальное.
568(8) — В головном модуле используется массив с регулируемыми размерами.
570(8) — Для оператора требуется, чтобы имя было именем массива, переменной или формальным параметром.
571(8) — В логическом выражении отсутствует или неправильно используется логическая операция.
572(8) — В логической константе отсутствует или неправильно используется точка.
573(8) — Для комплексного основания используется показатель степени не целого типа,
574(8) — Используется показатель степени комплексного типа.
575(8) — Выражение содержит операнды логического и не логического типов.
576(8) — Отношение содержит операнды логического или комплексного типа.
577(8) — Арифметический IF содержит операнды логического или комплексного типа.
578(8) — Тип фактического параметра не согласуется с типом формального параметра.
579(8) — Значение константы выходит за границы допустимой области.
580(8) — Литерал используется в недопустимом предложении.
583(8) — Переменная отсутствует или массив не объявлен.
584(8) — Выражение в объявлении внутренней функции содержит элементы массива.
585(8) — Элемент массива содержит неверное количество индексных выражений.
586(8) — Неправильное объявление COMMON в модуле BLOCK DATA.
587(8) — Логическая константа указана не как .TRUE. или .FALSE. .
588(8) — Используется переменная не целого типа.
589(8) — Переменная целого типа ранее была объявлена как массив.
590(8) — Тип переменной ранее объявлялся отличным от целого типа.
591(8) — Переменной целого типа присваивается начальное значение не в виде целой константы.
593(8) — Символ «&» используется не в операторе CALL.

594(8) — Неправильное использование апострофа.
595(8) — В объявлении FORMAT неправильно используется апостроф.
596(8) — Используемое в списке ввода-вывода имя не является именем массива или переменной.
597(8) — В параметре SUBCHK объявления DEBUG указано не имя массива.
598(8) — Объявление AT используется не в пакете отладки.
599(8) — В объявлении AT указана метка, не определенная в программном модуле.
600(8) — Оператор TRACE ON размещен перед объявлением AT.
601(8) — В операторе TRACE не указаны символы «ON» или «OFF».
602(8) — Программный модуль, который транслируется в режиме TEST, содержит предложения отладки.

Сообщения программы преобразования (префикс IGK)

700, 707, 708(4) — В задании отсутствует оператор DD, описывающий выводной (700) или вводной (707) набор данных или набор данных печати (708), либо в нем допущены ошибки.
701(4) — Произошла ошибка ввода-вывода.
703(0) — Вводной набор данных содержит запись, состоящую из пробелов.
704(0) — Выполнение программы закончено.
705(4) — Имеется ошибка в указании режимов преобразования.
706(0) — Основной памяти недостаточно для размещения комментариев.

Таблица 1

Сообщения посредника транслятора Фортран SE (префикс IHM)

Номер сообщения	Причина	Действие абонента
78101	Все наборы данных, описанные операторами DD с операндом DYNAM в процедуре LOGON, уже распределены	Освободить ненужные наборы данных и повторно ввести команду
78102	Набор данных отсутствует на томе, на который указывает запись каталога	Скорректировать запись в каталоге и повторно ввести команду
78103	Требуемый том не подготовлен к работе	Установить нужный том и повторно ввести команду
78104, 78108, 78164	Ошибка вычислительной системы	Повторно ввести команду
78105	Набор данных используется другим абонентом или заданием	Ввести команду позднее

Продолжение табл. 1

Номер сообщения	Причина	Действие абонента
78106	Раздел (указанный или TEMPNAME) отсутствует в библиотечном наборе данных	Ввести правильное имя
78107	Имя набора данных отсутствует в каталоге	Ввести правильное имя
78112	Неправильно составлено полностью уточненное имя	Повторно ввести команду, указав правильное имя
78116	Имя раздела указано для набора данных, который не является библиотечным	Ввести правильное имя (примечание)
78120	Длина полностью уточненного имени превышает 44 символа	Ввести правильное имя (примечание)
78124	На доступных абоненту томах недостаточно памяти для размещения указанного набора данных	Удалить ненужные наборы данных и повторно ввести команду
78132	Набор данных размещается на нескольких томах, что недопустимо в СРВ	Правильно разместить набор данных и повторно ввести команду

Примечание. Если ошибка допущена в имени набора данных печати или с объектным модулем, необходимо повторно ввести команду.

Таблица 2

Сообщения посредника транслятора Фортран СС (префикс IGK)

Номер сообщения	Причина	Действие абонента
101(201)	Все наборы данных, описанные операторами DD с операндом DYNAM в процедуре LOGON, уже распределены	Освободить ненужные наборы данных и повторно ввести команду
102(202)	Набор данных отсутствует на томе, на который указывает запись каталога	Скорректировать запись в каталоге и повторно ввести команду
103(203)	Требуемый том не подготовлен к работе	Установить нужный том и повторно ввести команду

Продолжение табл. 2

Номер сообщения	Причина	Действие абонента
104(204), 105(205), 108, 112(212), 113(213)	Ошибка вычислительной системы	Повторно ввести команду
106(206)	Набор данных используется другим абонентом или заданием	Ввести команду позднее
107	Длина полностью уточненного имени превышает 44 символа	Повторно ввести команду, указав правильное имя
109	Имя набора данных отсутствует в каталоге	Повторно ввести команду, указав правильное имя
110	Раздел (указанный или TEMPNAME) отсутствует в библиотечном наборе данных	Повторно ввести команду, указав правильное имя
111	Имя раздела указано для набора данных, который не является библиотечным	Повторно ввести команду, указав правильное имя
114(214)	На доступных абоненту томах недостаточно памяти для размещения указанного набора данных	Удалить ненужные наборы данных и повторно ввести команду
115(215)	Неправильно составлено полностью уточненное имя	Повторно ввести команду, указав правильное имя
132	Набор данных размещается на нескольких томах, что недопустимо в СРВ	Правильно разместить набор данных и повторно ввести команду

Таблица 3

Сообщения посредника программы преобразования (префикс IGK)

Номер сообщения	Причина	Действие
001, 006, 008, 300, 311	Ошибка вычислительной системы	Повторно ввести команду
002	Все наборы данных, описанные операторами DD с операндом DYNAM в процедуре LOGON, уже распределены	Освободить ненужные наборы данных и повторно ввести команду

Продолжение табл. 3

Номер сообщения	Причина	Действие
003	Требуемый том не подготовлен к работе	Установить нужный том и повторно ввести команду
004	Имя набора данных отсутствует в каталоге	Повторно ввести команду, указав правильное имя
005	Неправильно составлено полностью уточненное имя	Повторно ввести команду, указав правильное имя
010	Набор данных используется другим абонентом или заданием	Ввести команду позднее
012	Требуемый том не подготовлен к работе	Установить нужный том и повторно ввести команду
014	На доступных абоненту томах недостаточно памяти для размещения указанного набора данных	Удалить ненужные наборы данных и повторно ввести команду
016, 313	Длина полностью уточненного имени превышает 44 символа	Повторно ввести команду, указав правильное имя
019	Набор данных размещается на нескольких томах, что недопустимо в СРВ	Правильно разместить набор данных и повторно ввести команду
303	Для библиотечного набора данных не задан раздел	Повторно ввести команду, указав имя раздела
304	Указанный раздел отсутствует в библиотечном наборе данных	Повторно ввести команду, указав правильное имя
305	Организация набора данных отлична от библиотечной или последовательной	Повторно ввести команду, указав правильное имя
306	Указанный набор данных не является библиотечным	Повторно ввести команду, указав правильное имя
307	Набор данных, предназначенный для преобразованной программы, уже существует в системе	Повторно ввести команду, указав другое имя
315	В команде CONVERT пропущен операнд IN или OUT	Повторно ввести команду, устранив ошибку
316	В команде CONVERT не указан операнд FORTCC	Повторно ввести команду, устранив ошибку

Таблица 4

Сообщения посредника Диалогового отладчика (префикс ILM)

Номер сообщения	Причина	Действие
000	Требуется повторный ввод информации	1
001	Установлен режим ввода подкоманд	2
002	Требуется выполнить корректирующее действие	2
101	Все наборы данных, описанные операторами DD с операндом DYNAM в процедуре LOGON, уже распределены	3, или 4, или 6
102	Набор данных отсутствует на томе, на который указывает запись каталога	3, или 4, или 5, или 6
103	Требуемый том не подготовлен к работе	3, или 4, или 5, или 6
104, 108	Ошибка вычислительной системы	3, или 4, или 5, или 6
105	Набор данных используется другим абонентом или заданием	3, или 4, или 5, или 6
106	Раздел отсутствует в библиотечном наборе данных	3, или 6, или 7
107	Имя набора данных отсутствует в каталоге	3, или 6, или 7
112	Неправильно составлено полностью уточненное имя	7
116	Имя раздела указано для набора данных, который не является библиотечным	3, или 6, или 7
120	Длина полностью уточненного имени превышает 44 символа	3, или 5, или 7
124	На доступных абоненту томах недостаточно места для размещения набора данных печати	4 или 6
132	Набор данных размещается на нескольких томах, что недопустимо в СРВ	3, или 4, или 5, или 6
164	Ошибка вычислительной системы	6

Продолжение табл. 4

Номер сообщения	Причина	Действие
180	Не удалось автоматически распределить набор данных по одной из следующих причин: а) возникли затруднения при построении имени набора данных; б) имя набора данных отсутствует в каталоге	1, или 3, или 6
181	По требованию абонента распределено уже шесть наборов данных с исходной программой. Отладка продолжается без распределения указанного набора данных	Нет
182	Автоматически распределено уже шесть наборов данных с исходной программой. Отладка продолжается без распределения указанного набора данных	Нет

Действие 1. Ввести информацию, запрашиваемую предыдущим сообщением.

Действие 2. Ввести любую подкоманду.

Действие 3. Если набор данных содержит исходную программу и его можно не распределять, ввести пустую строку.

Действие 4. Если набор данных используется для печати и его можно не распределять, запросить прерывание по сигналу ВНИМАНИЕ. После этого ввести команду TESTFORT с операндом NOPRINT.

Действие 5. Если набор данных используется для печати и вместо него можно использовать набор данных с другим именем, ввести это имя.

Действие 6. Если набор данных используется для печати или содержит исходную программу и его необходимо распределить, запросить прерывание по сигналу ВНИМАНИЕ для того, чтобы вернуться в режим команд СРВ. Средствами операционной системы и СРВ проанализировать причину ошибки и устранить ее. После этого повторно ввести команду TESTFORT. Если ситуация повторяется, обратиться к системному программисту.

Действие 7. Если в имени допущена ошибка, ввести правильное имя.

Таблица 5

Сообщения диалогового отладчика (префикс ILM)

Номер сообщения	Причина	Действие
191,198, 999	Ошибка вычислительной системы	1 или 2
192	Невозможно открыть набор данных с исходной программой	Нет (примечание 1)
193	Ошибка ввода-вывода при обращении к набору данных с исходной программой	Нет (примечание 1)
194	Подкоманда SOURCE используется для программного модуля, который не доступен абоненту	3
195	В программе нет ни одного модуля, протранслированного в режиме TEST. Нужна трансляция в режиме TEST	—
196	Набор данных печати не доступен абоненту	Нет (примечание 2)
197	Невозможно открыть набор данных печати	Нет (примечание 2)
200	Недостаточно основной памяти для выполнения подкоманды. Упростить подкоманду	—
220	Ошибка ввода-вывода при обращении к набору данных с исходной программой или к набору данных печати	Нет (примечание 1)
225	Ошибка ввода-вывода при обращении к набору-данных печати	Нет (примечание 2)
240	Значение индексного выражения превосходит верхнюю границу измерения	3
245	Значение индексного выражения равно нулю или меньше нуля	3
290	Недостаточно основной памяти для вывода сообщения	1
291	Введена подкоманда LIST во время подготовки сообщения об ошибке в рабочей программе. Ввести подкоманду GO или RUN	—
300	Уведомление о безусловной точке прерывания	4
301	Уведомление о временной точке прерывания	4
303	Уведомление о входе в программный модуль	Нет
304	Уведомление о передаче управления внутри программного модуля	Нет
305	Уведомление о выходе из программного модуля	Нет
306	Нормальное окончание выполнения программы	5
307	Аварийное завершение	2
400	Выполнение списка подкоманд прекращено по сигналу ВНИМАНИЕ	Нет

Продолжение табл. 5

Номер сообщения	Причина	Действие
405	Подкоманда введена при подготовке сообщения об ошибке в рабочей программе	3 или 4
410	Неверное имя подкоманды	3
420	В подкоманде допущена ошибка	3
455	В подкоманде LIST допущена ошибка	6
460	Неправильное размещение скобок в подкоманде IF или в списке подкоманд подкоманды AT. Ввести информацию, исключая внешние скобки	—
470	Подкоманда IF или список подкоманд подкоманды AT содержит неправильную подкоманду	3
500	В подкоманде задана метка или номер строки невыполняемого предложения	7
510	Неверный диапазон номеров строк	7
520	В подкоманде QUALIFY задано имя программного модуля, не протранслированного в режиме TEST	7
525	Неправильное использование скобок в подкоманде FIXUP	6
530	В подкоманде FIXUP допущена ошибка	6
535	В подкоманде FIXUP задана логическая переменная или логическая константа	6
540	В подкоманде FIXUP используется недопустимый элемент Фортрана	6
545	В ответ на сообщение вместо элемента Фортрана введена пустая строка или в подкоманде LIST отсутствует левая граница диапазона	7
600	Неправильное использование скобок в элементе Фортрана	7
605	В комплексной константе используется целая константа	7
610	В комплексной константе используются вещественные константы разной длины	7
615	В элементе Фортрана допущена ошибка	7
620	Переменная задана с индексом	7
625	Количество индексных выражений не совпадает с размерностью массива	7
630	В имени переменной допущена ошибка	7
635	Имя переменной отсутствует в программе	7
640	В константе допущена ошибка	7
645	Значение константы выходит за границы допустимой области	7
650	В качестве индексного выражения используется имя элемента массива или массива	7
655	В комплексной константе отсутствует действительная либо мнимая часть	7

Продолжение табл. 5

Номер сообщения	Причина	Действие
660	В индексе элемента массива допущена ошибка	7
665	В индексном выражении используется не целая переменная	7
670	В подкоманде SET неправильно задана левая часть оператора присваивания	6
675	В подкоманде SET неправильно задана правая часть оператора присваивания	6
680	В подкоманде SET используются элементы логического и других типов	6
685	Неправильное использование скобок в подкоманде SET	6
690	В подкоманде LIST допущена ошибка	6
695	Неправильное использование скобок в подкоманде LIST	6
700	В подкоманде LIST отсутствует переменная	6
705	Имя переменной задано с неправильным программным уточнителем	7
710	Неправильное использование скобок в условии IF или WHEN	8
715	В условии WHEN используется не логическая переменная	8
720	В условии допущена ошибка	8
725	В условии заданы переменные разных типов	8
730	Нарушен синтаксис условия	8
735	В условии задан недопустимый элемент Фортрана	8
740	В подкоманде IF допущена ошибка	6
745	В подкоманде WHEN допущена ошибка	6
750	В подкоманде IF после условия отсутствует подкоманда	6
755	Подкоманда QUALIFY недопустима в подкоманде IF	6
760	Не задано условие в подкоманде IF или WHEN	6
765	В имени условия подкоманды WHEN допущена ошибка	6
770	Неправильное использование скобок в условии подкоманды WHEN или IF	6
775	Неправильное использование подкоманды HELP.	Нет
780	Подкоманда содержит лишний операнд	Нет
785	В подкоманде ERROR неправильно задан диапазон номеров ошибок	7
790	В подкоманде ERROR номер ошибки меньше 200 или больше максимально допустимого	7

Продолжение табл. 5

Номер сообщения	Причина	Действие
795	В подкоманде SET операнд логического типа используется вместе с операндом другого типа	6
800	В подкоманде GO указан оператор, не принадлежащий программному модулю	3
801	Уведомление о значении текущего программного уточнителя	Нет
802	Подкоманда GO или RUN введена после нормального завершения программы	5
840	Строка вывода по подкоманде LIST без операнда DUMP	Нет
850	Строка вывода по подкоманде LIST с операндом DUMP	Нет
860	В подкоманде FIXUP для элемента массива указано неправильное индексное выражение	3
861	Неправильное использование подкоманды FIXUP	4
862	Возникла ошибка, для которой можно выполнить только стандартное корректирующее действие. Ввести подкоманду GO	—
863	В подкоманде FIXUP заданы два операнда, а должен быть задан один	3
864	Делается попытка по подкоманде FIXUP исправить ошибку с номерами 211, 215 или 225, что недопустимо. Ввести подкоманды NEXT и GO, а затем скорректировать данное по подкоманде SET	—
865	В случае ошибки введена подкоманда GO или RUN с операндом, что недопустимо	3
866	Подкоманда введена после прерывания по сигналу ВНИМАНИЕ во время обработки ошибки	3 или 4
867	Уведомление об ошибке	Нет
870	Уведомление об условной точке прерывания при входе в программный модуль	4
871	Уведомление об условной точке прерывания внутри программного модуля	4
875	В подкоманде WHEN задано имя условия, которое не определено	4
876	В условии WHEN задан неправильный индекс	3 или 4
880,881	В подкоманде OFFWN заданы имена условий, которые не являются активными	4
920	Это сообщение следует за сообщением ILM245I, если неправильное индексное выражение задано в подкоманде SET	3 или 4

Продолжение табл. 5

Номер сообщения	Причина	Действие
925	В подкоманде SET количество данных превышает размерность массива. Лишние данные игнорируются	4
930	Заголовок к исходной программе, выводимой по подкоманде SOURCE	Нет
931	В наборе данных с исходной программой формат записей отличен от F или V	4 (примечание 3)
932	В наборе данных с исходной программой обнаружена неверная запись	4 (примечание 3)
933	Невозможно открыть набор данных с исходной программой	4 (примечание 3)
934	Ошибка ввода-вывода при обращении к набору данных с исходной программой	4 (примечание 3)
935	Содержимое строки исходной программы	Нет
937	В подкоманде SOURCE номер строки задан неправильно	3
938	В подкоманде SOURCE диапазон строк задан неправильно	3
940	Заголовок списка операторов, которые не выполнялись	Нет
942	Строка списка операторов, которые не выполнялись	Нет
943	Уведомление об отсутствии операторов, которые не выполнялись, или об отсутствии безусловных точек прерывания или условий WHEN	Нет
945	Заголовок списка операторов, содержащего сведения о частоте их выполнения	Нет
947	Строка списка операторов, содержащего сведения о частоте их выполнения	Нет
950	Заголовок списка безусловных точек прерывания	Нет
952	Строка списка безусловных точек прерывания	Нет
953	Строка списка безусловных точек прерывания с указанием частоты их установки	Нет
955	Заголовок списка условий WHEN	Нет
957	Строка списка условий WHEN	Нет
990,992	Подкоманда WHERE введена до начала выполнения программы или тогда, когда не выполнялся ни один модуль, протранслированный в режиме TEST	4
991	Уведомление о передаче управления между программными модулями	Нет
995	Уведомление о местоположении текущего оператора	Нет
996	Уведомление о передаче управления внутри программного модуля. За ним всегда следует сообщение с номером 997	Нет

Продолжение табл. 5

Номер сообщения	Причина	Действие
997	Сообщение всегда следует за сообщением с номером 996. Эти сообщения информируют о передаче управления от оператора, указанного в сообщении с номером 996, к оператору, указанному в сообщении с номером 997	Нет

Действие 1. Повторно ввести команду TESTFORT.

Действие 2. Известить системного программиста.

Действие 3. Повторно ввести подкоманду, устранив в ней ошибки, если требуется.

Действие 4. Ввести любую подкоманду.

Действие 5. Ввести любую подкоманду, не возобновляющую выполнение программы.

Действие 6. Ввести подкоманду, опустив ее имя.

Действие 7. Ввести элемент, в котором допущена ошибка, включая счетчик повторения, если он требуется. Элементом может быть метка оператора, номер строки оператора, константа, имя переменной, имя массива и т. д.

Действие 8. Ввести условие IF или WHEN, исключая внешние скобки.

Примечания.

1. Набор данных становится недоступным для абонента.
2. Отладочная информация выводится на АП.
3. Не следует использовать подкоманду SOURCE в сеансе отладки для указанного набора данных.

Приложение 2

ФОРМАТЫ КОМАНД И ПОДКОМАНД СРВ

В табл. 1—3 приводятся форматы команд и подкоманд СРВ, которые обеспечивают работу с транслляторами Фортран СС и Фортран SE. При вводе команд (подкоманд) допускается использование сокращений для ключевых слов ключевых операндов. Достаточно ввести такую часть ключевого слова, которая позволяла бы системе отличить один операнд команды от других операндов этой же команды. Например, используя сокращения, команду ALLOCATE можно ввести в виде:

```
alloc f(prog) d(name) n v(lab560) sp(10,5) b(400)
```

или

```
alloc fi(prog) da(name) ne vo(lab560) spac(10,5) bl(400)
```

Таблица 1

Форматы команд

Имя команды	Операнды
{ALLOCATE / ALLOC}	{FILE(имя) [DATASET ({* / имя набора данных})] / DATASET ({* / имя набора данных}) [FILE(имя)]} [OLD / SHR / MOD / NEW / SYSOUT] [VOLUME(регистрационный номер)] [SPACE(количество [, приращение]) BLOCK(длина) [DIR(число)]]
CALL	имя набора данных
{CONVERT / CON}	имя набора данных OUT(имя набора данных) FORTCC [FREE / FIXED]
{DELETE / D}	список имен наборов данных [PURGE / NOPURGE]
{EDIT / E}	имя набора данных [NEW / OLD]
	[FORTSE / FORTCC [({FREE / FIXED})] / DATA] [SCAN / NOSCAN] [NUM / NONUM] [BLOCK(число)] [LINE(число)]
FORTCC	имя набора данных [FIXED / FREE] [LMSG / SMSG] [TEST]

Продолжение табл. 1

Имя команды	Операнды
FORTSE	имя набора данных [PRINT / NOPRINT] [(*/имя набора данных)] [ID / NOID] [LINECNT(число)] [LIST / NOLIST] [LOAD / NOLOAD] [(имя набора данных)] [MAP / NOMAP] [NAME(имя модуля)] [SOURCE / NOSOURCE] [TERM / NOTERM] [TEST]
FREE	{DATASET (список имен наборов данных) / FILE (список имен)}
{HELP / H}	[имя команды [FUNCTION] [SYNTAX] [OPERANDS[(список операндов)]] [ALL]]
LINK	(список имен наборов данных) FORTLIB [LIB (список имен наборов данных)] [LOAD[(имя набора данных)]] [PRINT [(*/имя набора данных)] / NOPRINT] [список режимов]
{LISTALC / LISTA}	[STATUS] [HISTORY] [MEMBERS] [SYSNAMES]
{LISTCAT / LISTC}	[HISTORY] [VOLUMES] [MEMBERS] [LEVEL(индекс)]
{LISTDS / LISTD}	(список имен наборов данных) [STATUS] [HISTORY] [MEMBERS] [LABEL]
{LOADGO / LOAD}	(список имен наборов данных) FORTLIB [LIB (список имен наборов данных)] [PRINT [(*/имя набора данных)] / NOPRINT] [список режимов]

Продолжение табл. 1

Имя команды	Операнды
LOGOFF	
LOGON	идентификатор абонента[/пароль] [ACCT(учетный номер)][PROC(имя процедуры)]
[PROFILE PROF]	[CHAR(символ) NOCHAR] [LINE(символ) NOLINE] [PROMPT NOPROMPT] [INTERCOM NOINTERCOM] [MSGID NOMSGID]
{RENAME REN}	старое имя новое имя
{RUN R}	имя набора данных {FORTCC FORTSE [FIXED FREE] [LMSG SMSG]}
SEND	'текст' [USER(список идентификаторов) OPERATOR[(число)] [NOW LOGON]]
{TERMINAL TERM}	[LINES (число) NOLINES] [SECONDS(число) NOSECONDS] [INPUT(строка символов) NOINPUT]
{TESTFORT TESTE}	(список имен наборов данных) [LIB(список имен наборов данных)] [SOURCE(список имен наборов данных) NOSOURCE] [PRINT[(имя набора данных)] NOPRINT] [RES NORES]

Таблица 2

Подкоманды команды EDIT

Имя подкоманды	Операнды
{CHANGE / C}	номер строки [номер строки] ХцепочкаХцепочка[X[ALL]]
{CHANGE / C}	номер строки [номер строки] {Хцепочка / счетчик}
{DELETE / D}	номер строки [номер строки]
END	
{HELP / H}	[имя подкоманды[FUNCTION] [SYNTAX] [OPERANDS[(список операндов)]] [ALL]]
{INPUT / I}	[номер строки [шаг]] [I / R]
{LIST / L}	[номер строки [номер строки]]
{RENUM / REN}	[новый номер строки [шаг[старый номер строки]]]
{RUN / R}	[LMSG / SMSG]
{SAVE / S}	[имя набора данных]
{SCAN / SC}	[номер строки [номер строки] ON OFF]

Таблица 3

Подкоманды команды TESTFORT

Имя подкоманды	Операнды
AT	{идентификатор оператора идентификатор оператора: идентификатор оператора (список идентификаторов операторов)} [(список подкоманд)] [COUNT(n)][NOTIFY / NONOTIFY]
END	
{ERROR ER}	{номер ошибки номер ошибки:номер ошибки (список номеров ошибок)} [MSG / NOMSG] [EXIT / NOEXIT]
{FIXUP F}	[ARG1(значение)] [ARG2(значение)]
GO	[идентификатор оператора]
{HALT HA}	
{HELP H}	[имя подкоманды [FUNCTION] [SYNTAX] [OPERANDS[(список операндов)]] [ALL]]
IF	(условие) подкоманда
{LIST L}	{имя имя : имя (список имен) *} [DUMP[(код)]] [PRINT]
LISTBRKS	
LISTFREQ	[идентификатор оператора идентификатор оператора: идентификатор оператора (список идентификаторов операторов)] [ZEROFREQ] [PRINT]
{NEXT N}	

Продолжение табл. 3

Имя подкоманды	Операнды
OFF	[идентификатор оператора идентификатор оператора: идентификатор оператора (список идентификаторов операторов)]
OFFWN	[имя условия (список имен условий)]
PURGE	
{QUALIFY Q}	[имя программного модуля]
{RUN R}	[идентификатор оператора]
{SET S}	оператор присваивания
{SOURCE SO}	[идентификатор оператора идентификатор оператора: идентификатор оператора (список идентификаторов операторов) *]
{TRACE T}	[STMT ENTRY OFF] [PRINT]
{WHEN WN}	имя условия [условие]
{WHERE W}	[TRBACK NOTRBACK] [FLOW] [PRINT]

ОГЛАВЛЕНИЕ